ELEX MEDIA KOMPUTINDO

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional 3

Agustinus Nalwan

Disertai DISKET

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional 3

Agustinus Nalwan

Seri Aplikasi Pemrograman

Pemrograman Animasi dan Game Profesional 3

3D Virtual Reality Game dan Graphic Programming

Sanksi Pelanggaran Pasal 44:
Undang-undang Nomor 7 Tahun 1987 Tentang
Perubahan atas undang-undang Nomor 6 Tahun 1982
Tentang Hak Cipta

1. Barangsiapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu ciptaan atau memberi izin untuk itu, dipidana dengan pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 100.000.000,- (seratus juta rupiah).

2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 50.000.000,- (lima puluh juta rupiah).

Seri Aplikasi Pemrograman

Pemrograman Animasi dan Game Profesional 3

3D Virtual Reality Game
dan
Graphic Programming

Agustinus Nalwan

Penerbit PT Elex Media Komputindo
Kelompok Gramedia-Jakarta

Seri Aplikasi Pemrograman
Pemrograman Animasi dan Game Profesional 3
3D Virtual Reality Game dan Graphic Programming
Agustinus Nalwan
© 1996, PT Elex Media Komputindo, Jakarta
Hak cipta dilindungi undang-undang
Diterbitkan pertama kali oleh
PT Elex Media Komputindo
Kelompok Gramedia, Anggota IKAPI, Jakarta 1996

12196786
ISBN 979-663-358-2

Dilarang keras menerjemahkan, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari Penerbit.

Dicetak oleh Percetakan PT Gramedia, Jakarta
Isi di luar tanggung jawab percetakan

Pendahuluan

Dunia pemrograman game makin berkembang pesat dengan didukung oleh processor yang semakin cepat dan disertai dengan munculnya hardware-hardware baru yang mendukung graphic tiga dimensi. Sehingga banyak bermunculan game-game, seperti DUKE NUKEM 3D, QUAKE dan lain-lain.

Kita juga tidak mau kalah, pada kesempatan ini kita akan membahas 3D PROGRAMMING!!! Di sini akan dijelaskan bagaimana merancang sebuah FLIGHT SIMULATOR yang dudukung dengan teknik pencahayaan (Light Sourcing), teknik-teknik transformasi dan filter-filter tiga dimensi. Pada bagian terakhir akan dijelaskan bagaimana meng-optimize (mempercepat) sebuah game tiga dimensi sehingga memiliki kecepatan yang memadai untuk dimainkan.

Pembahasan ini adalah dasar dari pemrograman tiga dimensi dan akan sangat membantu pembahasan-pembahasan game tiga dimensi kita pada edisi mendatang.

Compiller yang digunakan tetap Turbo Pascal 7.0 yang dibantu dengan routine Assembler.

Untuk lebih jelasnya, lanjutkan dengan membaca bagian pembahasan.

Daftar Isi

BAB 1 PENGENALAN TIGA DIMENSI **1**

1.1 PENDAHULUAN 1
1.2 DASAR-DASAR TIGA DIMENSI 2
1.2.1 Parallel Projection *3*
1.2.2 Perspective Projection *11*
1.3 DASAR-DASAR GEOMETRI DAN TRANSFORMASI TIGA DIMENSI 19
1.3.1 Coordinate System *19*
1.3.2 Coordinate Plane *20*
1.3.3 Movement *22*

BAB 2 PROYEKSI OBJEK SECARA BITMAP **27**

2.1 PENDAHULUAN 27
2.2 SPRITE BANK ROTATION TECHNIQUE 28
2.3 PENGEMBANGAN SPRITE BANK ROTATION TECHNIQUE 31

BAB 3 PROYEKSI OBJEK SECARA VEKTOR **33**

3.1 3D VECTOR OBJEK DATA STRUCTURE 33
3.2 TRANSFORMASI 46
3.3 MATRIX 44
3.3.1 Definisi Matrix *44*
3.3.2 Operasi Matrix *45*
3.3.3 Matrix Identitas *46*
3.3.4 Transformasi *46*
3.3.5 Inverse Matrix *51*
3.3.6 Matrix Concatenation *52*
3.4 VEKTOR 54
3.4.1 Magnitude Vektor *54*
3.4.2 Penambahan dan Pengurangan Vektor *55*
3.4.3 Unit Vektor *55*
3.4.4 Dot Product *56*
3.4.5 Cross Product *57*
3.5 PERAKITAN UNIT MATH.TPU 58

BAB 4 LOCAL VS GLOBAL COORDINATE SYSTEM TRANSFORMATION **85**

4.1 LOCAL DAN GLOBAL COORDINATE SYSTEM 85

BAB 5 POLYGON SHADING PROJECTION **95**

5.1 POLYGON SHADING DATA STRUCTURE 95
5.2 POLYGON SHADING 116

BAB 6 3D FILTERING **139**

6.1 BACKFACE FILTERING 139
6.2 CROSS PRODUCT FILTERING 140
6.3 VERTEX ORDER FILTERING 155
6.4 DEPTH SORTING 161
6.5 VIEW VOLUME FILTERING 178

BAB 7 LIGHT SOURCING **195**

7.1 PENDAHULUAN 195

BAB 8 3D INVERSE TRANSFORMATION **211**

8.1 PENDAHULUAN 211
8.2 COORDINATE FRAME 235

BAB 9 OPTIMASI KECEPATAN **279**

9.1 PENDAHULUAN 279
9.2 OPTIMASI FLIGHT SIMULATOR 280

BAB 10 ULASAN TENTANG VGACAD.PAS **317**

10.1 DAFTAR FUNGSI DAN PROCEDURE DARI VGACAD.PAS 317

Penutup **328**

Informasi Daftar File

Program : B3PRL1.PAS
Description: Program proyeksi Titik dengan Parallel Projection.

Program : B3PRL2.PAS
Description: Program proyeksi Balok dengan Parallel Projection.

Program : B3PSV1.PAS
Description: Program proyeksi Titik dengan Perspective Projection.

Program : B3PSV2.PAS
Description: Program proyeksi Balok dengan Perspective Projection.

Program : B3VEC1.PAS
Description: Merancang Object dengan teknik vektor.

Program : MATH.PAS
Description: Unit matrix dan vektor transformasi.

Program : MATH3.PAS
Description: Unit Matrix dan Vektor yang telah di-optimize.

Program : B3LOC.PAS
Description: Program transformasi objek dengan teknik local coordinate system transformation.

Program : B3PLIST.PAS
Description: Program proyeksi wireframe dengan mempergunakan poligon shading data structure.

Program : B3SHADE.PAS
Description: Program transformasi objek dengan teknik Polygon Shading.

Program : B3SHADE3.PAS
Description: Program transformasi objek dengan filter BackFace normal calculation.

Program : B3SHADE5.PAS
Description: Program transformasi objek dengan filter Painter Sort.

Program : B3SHADE6.PAS
Description: Program transformasi objek dengan filter View Volume.

Program : B3SHADE7.PAS
Description: Program transformasi objek dengan flat shadded ambient light sourcing.

Program : B3VIEW.PAS
Description: Program proyeksi dengan menggunakan sudut pandang camera.

Program : B3FLGHT.PAS
Description: Program Flight Simulator dengan enam kebebasan gerak.

Program : B3OPTMZ.PAS
Description: Program Flight Simulator yang telah di-optimize.

Program : VGACAD.PAS
Description: Unit graphic mode 320 x 200 256 warna.

Program : VGAFONT.TPU
Description: Unit penggunaan font pada mode 320 x 200 256 warna.

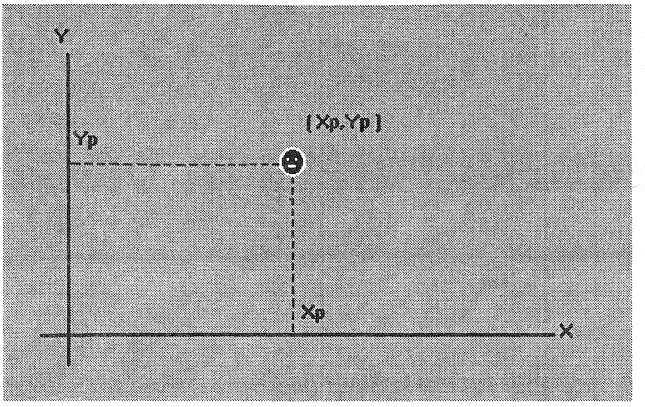
BAB 1
PENGENALAN TIGA DIMENSI

1.1 Pendahuluan

Berawal dari meledaknya game DOOM di pasaran mengakibatkan para game developer bersaing menciptakan game-game tiga dimensi. Memang sudah saatnya dunia game komputer beranjak ke game tiga dimensi dengan didukung oleh processor baru pentium yang sangat efektif untuk kalkulasi perhitungan tiga dimensi yang dulunya masih menjadi hambatan.

Nah sebenarnya apa sih yang menjadi perbedaan antara game dua dimensi dan game tiga dimensi? Tentunya bagi Anda penggemar game pastilah pernah bermain game simulator seperti F-15, F-19, Wing Commander, DOOM, juga game bertipe action-adventure seperti Prince of Persia, Sonic. Game-game F-15, F-19, Wing Commander dan DOOM adalah game tiga dimensi di mana kita dapat bergerak bebas ke atas, bawah, kiri, kanan, maju, mundur, berputar, pokoknya seakan-akan kita sedang berada di dalam dunia game yang kita mainkan tersebut dengan monitor sebagai mata kita. Sekarang bandingkan dengan group game kedua di atas yaitu Prince of Persia dan Sonic, di sini kita bertindak sebagai orang ketiga di mana kita melihat pemain yang sedang kita gerakkan dari sudut pandang samping.

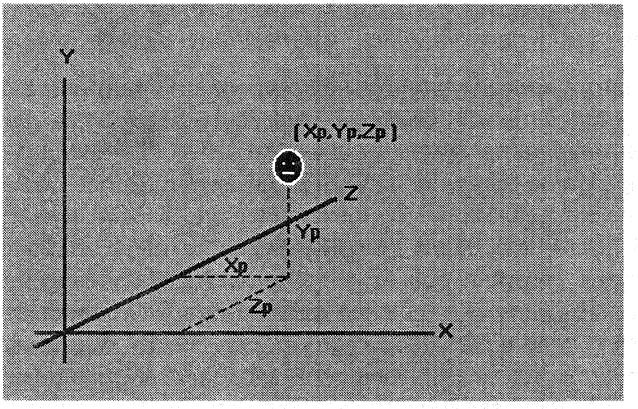
Tentu saja masing-masing mempunyai kelemahan dan kelebihan. Game tiga dimensi memberikan kita kebebasan lebih untuk menggerakkan pemain tetapi kadang-kadang karena kebebasan yang berlebihan ini dapat mengakibatkan permainan menjadi lebih rumit seperti terlalu banyaknya tombol yang diperlukan untuk mengontrol pemain.



Pada dunia dua dimensi, objek hanya memiliki dua axis yaitu X dan Y. Posisi objek ialah pada koordinat (Xp,Yp).

1.2 Dasar-dasar Tiga Dimensi

Dunia tiga dimensi tidaklah sesederhana dunia dua dimensi di mana kita hanya memiliki koordinat X dan Y saja melainkan di sini kita memiliki tiga buah koordinat/axis yaitu X, Y dan Z. Axis X adalah axis horizontal (posisi), axis Y adalah axis vertikal (ketinggian) sedangkan axis Z adalah axis yang menembus layar monitor Anda (kedalaman).



Pada dunia tiga dimensi kita memiliki tiga axis yaitu X, Y dan Z. Perhatikan objek pada gambar di atas terletak pada koordinat (Xp,Yp,Zp).

Nah sekarang yang menjadi masalah ialah, layar monitor kita adalah layar dua dimensi, bagaimana kita dapat menampilkan game tiga dimensi pada layar dua dimensi ini. Kelihatannya memang tidak masuk akal. Pernahkah

Anda memikirkan hal ini pada saat menonton film di bioskop maupun di televisi. Layar yang dipergunakan dua dimensi tetapi kok bisa menampilkan film sehingga kelihatan seperti tiga dimensi. Berarti hal yang kelihatannya tidak masuk akal itu pasti bisa dilakukan! Caranya ialah dengan melakukan 3D Projection (memproyeksikan koordinat tiga dimensi X,Y,Z dari suatu objek sehingga menjadi koordinat dua dimensi X dan Y pada layar monitor).

1.2.1 Parallel Projection

Teknik 3D Projection yang pertama yang akan kita pelajari ialah PARALLEL PROJECTION yang dapat diartikan sebagai proyeksi sejajar. Ini adalah sistem proyeksi yang paling sederhana yang biasanya dipergunakan pada pelajaran geometri SMA. Di sini kita hanya perlu mengabaikan koordinat Z dari suatu benda untuk memproyeksikan letaknya ke layar monitor. Jadi koordinat dua dimensi layar monitor yang dihasilkan ialah (Xs,Ys) di mana Xs dan Ys adalah koordinat tiga dimensi X dan Y dari objek. Mudah kan?

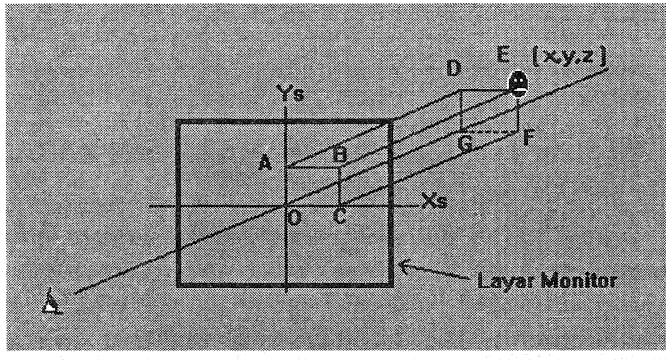


Diagram ini menggambarkan teknik parallel projection secara geometri.

Letak objek pada dunia tiga dimensi ialah (X,Y,Z) yaitu pada titik E. Sedangkan persegi panjang dengan garis tebal ialah layar monitor kita. Untuk memproyeksikan posisi benda dari sistem tiga dimensi ke dua dimensi dengan menggunakan parallel projection, kita cukup menarik garis lurus sejajar dengan sumbu Z (garis berawal mata yang menembus ke layar monitor) sehingga menembus layar monitor pada titik B. Di titik B inilah letak benda pada sistem dua dimensi. Satu hal lagi yang perlu diingat ialah mulai saat ini koordinat (0,0) layar berada di tengah layar dan karena kita menggunakan resolusi 320 x 200, maka koordinat (0,0) berada di koordinat layar (160,100).

Nah, kita telah mempelajarinya secara geometris, bagaimana kita menerapkannya secara matematis? Mudah, pada dunia tiga dimensi dari objek, koordinat X dari objek sebenarnya ialah panjang garis DE, koordinat Y objek ialah panjang garis EF sedangkan koordinat Z objek adalah panjang

garis BE. Di lain pihak, pada dunia dua dimensi hasil proyeksi, panjang garis AB ialah koordinat X sedangkan panjang garis BC adalah koordinat Y. Karena DE sejajar dan sama panjang dengan AB, EF juga sejajar dan sama panjang dengan BC maka koordinat dua dimensi X dan Y hasil proyeksi sama persis dengan koordinat tiga dimensi X dan Y objek.

Waktu untuk memprogram dimulai!!! Cobalah jalankan program di bawah ini.

```
{-------------------------------------------                    }
{ Program     : B3PRL1.PAS                                      }
{ Coder       : Agustinus Nalwan                                }
{ Description : Program proyeksi Titik dengan Parallel Projection }
{-------------------------------------------                    }

Uses Crt,Dos,VgaCad,VgaFont;

Const View_Distance = 400;

Var X,Y,Z:Real;
    Buff :Pointer;

Procedure Small;External;
{$L Small.Obj}

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Initialize;
  Begin
    OpenVga256;
    LoadFont(@Small);
    GetMem(Buff,64000);
    X:=0;Y:=0;Z:=600;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Function Strig(Num:Longint):String;
Var St:String;
  Begin
    Str(Num,St);
    Strig:=St;
  End;

Procedure Main;
Var Ch              :Char;
    pX,pY           :Longint;
  Begin
    Repeat
      Repeat
```

```
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        Pll_Projection(X,Y,Z,pX,pY);
        PutPixel(Buff^,pX,pY,13);
        PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
        PutString(Buff^,60,10,0,15,'Y : '+Strig(Round(Y)));
        PutString(Buff^,110,10,0,15,'Z : '+Strig(Round(Z)));
        MoveTo(Buff^,Screen^);
        Case Ch of
          #72:Y:=Y+2;
          #80:Y:=Y-2;
          #75:X:=X-2;
          #77:X:=X+2;
          '+','=':Z:=Z+2;
          '-','_':Z:=Z-2;
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Pada saat program ini Anda jalankan, akan terlihat garis horizontal dan vertikal yang membagi layar monitor menjadi empat bagian. Ini adalah sumbu X dan sumbu Y dari dunia tiga dimensi kita. Letak origin atau titik (0,0,0) ialah di tengah-tengah layar. Perhatikan sebuah titik kecil di tengah-tengah monitor. Ini adalah objek tiga dimensi kita yang pertama kali. Pada kiri atas layar terdapat status koordinat X, Y dan Z dari titik tersebut. Cobalah gerakkan titik tersebut.

Tombol-tombol fungsi:

Kursor Kiri	Mengurangi koordinat X
Kursor Kanan	Menambah koordinat X
Kursor Atas	Menambah koordinat Y
Kursor Bawah	Mengurangi koordinat Y
Tombol +	Menambah koordinat Z
Tombol -	Mengurangi koordinat Z
ESC	Keluar
Tombol Lain	Pause

Karena teknik proyeksi titik yang kita gunakan ialah PARALLEL PROJECTION, maka pada saat koordinat Z kita ubah, hasil proyeksi tidak

akan berpindah sebagaimana yang kita ketahui bahwa PARALLEL PROJECTION mengabaikan koordinat Z.

PENJELASAN LISTING PROGRAM

Kita tidak akan membahas listing program secara detil karena format program yang dipakai hampir sama dengan apa yang telah kita pelajari seperti animasi tiga layar, fungsi-fungsi VGACAD dan lain-lainnya.

Ok, kita mulai dengan bermain program.

```
Procedure Main;
Var Ch          :Char;
```

pX dan pY adalah variabel yang akan kita gunakan untuk menyimpan koordinat dua dimensi hasil proyeksi.

```
    pX,pY           :Longint;
  Begin
    Repeat
      Repeat
```

Tampilkan, garis koordinat X dan Y.

```
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
```

Lakukan parallel projection terhadap koordinat tiga dimensi X, Y, Z dari titik.

```
    Pll_Projection(X,Y,Z,pX,pY);
```

Tampilkan titik pada layar monitor.

```
    PutPixel(Buff^,pX,pY,13);
```

Tampilkan koordinat titik kemudian lakukan pemindahan virtual screen ke visual screen.

```
        PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
        PutString(Buff^,60,10,0,15,'Y : '+Strig(Round(Y)));
        PutString(Buff^,110,10,0,15,'Z : '+Strig(Round(Z)));
        MoveTo(Buff^,Screen^);
```

Deteksi tombol keyboard untuk mengubah koordinat tiga dimensi X,Y,Z dari titik.

```
        Case Ch of
          #72:Y:=Y+2;
          #80:Y:=Y-2;
          #75:X:=X-2;
          #77:X:=X+2;
          '+','=':Z:=Z+2;
          '-','_':Z:=Z-2;
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;
```

Sekarang marilah kita menyelidiki Parallel Projection pada program di atas.

```
Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
```

Ambil koordinat X yang ditambah dengan 160 (koordinat X tengah layar) sebagai hasil koordinat dua dimensi X transformasi. Untuk koordinat Y dilakukan 100-Round(Y) karena pada dunia tiga dimensi, koordinat Y positif ialah ke arah atas, sedangkan pada koordinat layar monitor, koordinat Y positif ialah ke arah bawah.

```
    pX:=160+Round(X);pY:=100-Round(Y);
  End;
```

Sejauh ini, kita masih bermain-main dengan titik. Sekarang kita akan mencoba memproyeksikan sebuah objek yang memiliki panjang dan lebar. Kita ambil persegi panjang sebagai bahan eksperiment.

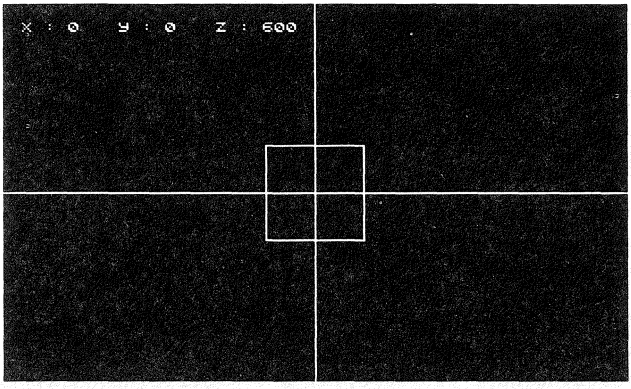
Cobalah jalankan program di bawah ini.

```
{---------------------------------------------                    }
{ Program     : B3PRL2.PAS                                       }
{ Coder       : Agustinus Nalwan                                 }
{ Description : Program proyeksi Balok dengan Parallel Projection }
{---------------------------------------------                    }

Uses Crt,Dos,VgaCad,VgaFont;

Const View_Distance = 400;

Var X,Y,Z:Real;
    Buff :Pointer;
    Width,Length:Longint;


Procedure Small;External;
{$L Small.Obj}

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;


Procedure Initialize;
  Begin
    OpenVga256;
    LoadFont(@Small);
    GetMem(Buff,64000);
    X:=0;Y:=0;Z:=600;
    Length:=50;Width:=50;
  End;


Procedure DeInitialize;
  Begin
    CloseVga256;
  End;
```

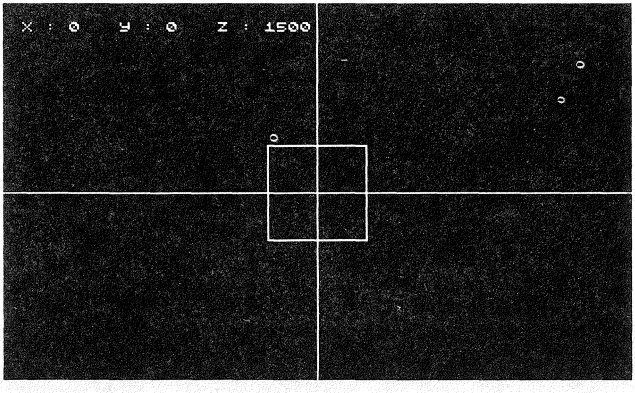
```
Function Strig(Num:Longint):String;
Var St:String;
  Begin
    Str(Num,St);
    Strig:=St;
  End;

Procedure Main;
Var Ch              :Char;
    pX1,pY1,pX2,pY2:Longint;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        Pll_Projection(X-Length div 2,Y+Width div 2,Z,pX1,pY1);
        Pll_Projection(X+Length div 2,Y-Width div 2,Z,pX2,pY2);
        Rectangle(Buff^,pX1,pY1,pX2,pY2,15);
        PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
        PutString(Buff^,60,10,0,15,'Y : '+Strig(Round(Y)));
        PutString(Buff^,110,10,0,15,'Z : '+Strig(Round(Z)));
        MoveTo(Buff^,Screen^);
        Case Ch of
          #72:Y:=Y+2;
          #80:Y:=Y-2;
          #75:X:=X-2;
          #77:X:=X+2;
          '+','=':Z:=Z+2;
          '-','_':Z:=Z-2;
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Di samping hanya menggerakkan sebuah titik pada layar monitor, sekarang kita mampu menggerakkan sebuah persegi panjang pada dunia tiga dimensi. Coba gerakkan persegi panjang tersebut dengan tombol fungsi yang sama dengan program kita yang terdahulu. Juga gerakkan koordinat Z. Hmmm ada yang aneh? Memang, walaupun persegi panjang tersebut ditambah koordinat Z-nya, tetapi ukurannya tetap sama. Padahal kenyataanya, jika sebuah objek digerakkan menjauhi si penglihat, objek tersebut harus mengecil. Sebenarnya apa yang terjadi di sini? Kejadian ini akan segera kita bahas.



Hasil proyeksi kotak yang berada pada koordinat Z=600.



Hasil proyeksi kotak pada koordinat Z=1500 masih tetap sebesar semula.

PENJELASAN LISTING PROGRAM

Perhatikan procedure Main di bawah ini.

```
Procedure Main;
Var Ch          :Char;
```

Karena persegi panjang yang akan kita proyeksikan dibentuk dari dua buah titik informasi yaitu titik kiri atas dan titik kanan bawah, maka diperlukan dua buah titik pula sebagai hasil transformasi.

```
    pX1,pY1,pX2,pY2:Longint;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
```

Dilakukan dua kali proyeksi. Yang pertama ialah proyeksi titik kiri atas dari persegi panjang dan yang kedua ialah proyeksi titik kanan bawah dari persegi panjang.

Karena proyeksi keduanya menggunakan PARALLEL PROJECTION, maka dengan mengabaikan koordinat Z, hasil proyeksi akan memiliki ukuran yang sama walaupun kita menggerakkan koordinat Z.

```
Pll_Projection(X-Length div 2,Y+Width div 2,Z,pX1,pY1);
Pll_Projection(X+Length div 2,Y-Width div 2,Z,pX2,pY2);
```

Tampilkan persegi panjang hasil proyeksi pada layar monitor.

```
Rectangle(Buff^,pX1,pY1,pX2,pY2,15);
PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
PutString(Buff^,60,10,0,15,'Y : '+Strig(Round(Y)));
PutString(Buff^,110,10,0,15,'Z : '+Strig(Round(Z)));
MoveTo(Buff^,Screen^);
Case Ch of
  #72:Y:=Y+2;
  #80:Y:=Y-2;
  #75:X:=X-2;
  #77:X:=X+2;
  '+','=':Z:=Z+2;
  '-','_':Z:=Z-2;
End;
Until Keypressed;
Ch:=Readkey;
Until Ch=#27;
End;
```

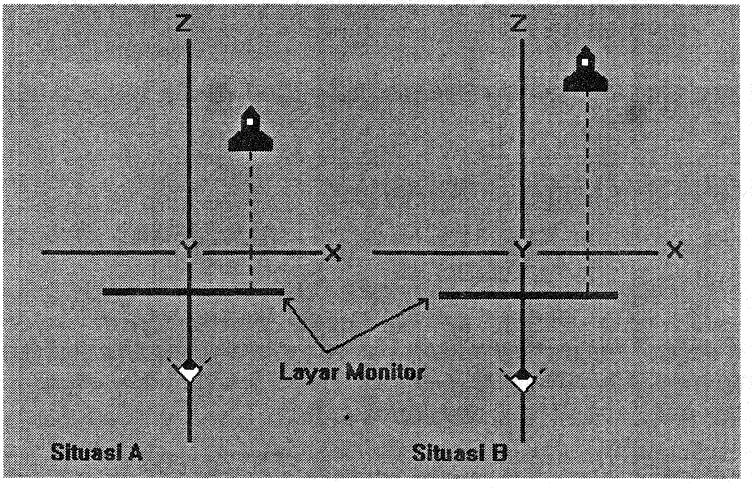


Diagram di atas menunjukkan dua lokasi objek yang berbeda dalam dunia tiga dimensi dilihat dari sudut pandang atas.

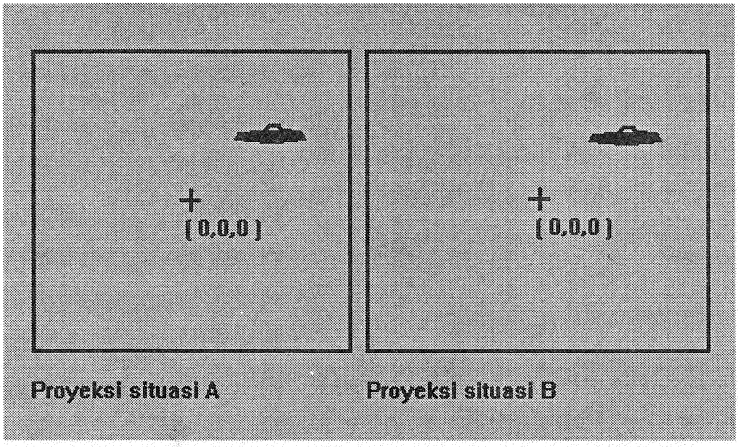


Diagram di atas menunjukkan hasil proyeksi dari dua lokasi objek yang berbeda pada layar monitor.

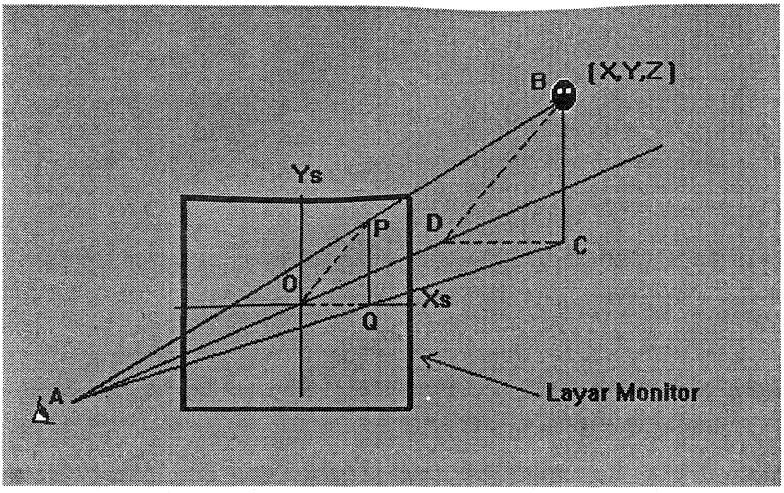
Perhatikan kedua gambar di atas. Gambar pertama yang terdiri atas sebuah objek pada situasi A dan pada situasi B di mana koordinat Z dari objek lebih jauh dibanding pada situasi A sedangkan koordinat X dan Y keduanya sama. Tetapi karena sistem proyeksi yang dipergunakan ialah parallel projection di mana kita mengabaikan koordinat Z dan menggunakan koordinat X dan Y objek sebagai koordinat hasil proyeksi, maka hasil proyeksi situasi A dan situasi B (pada gambar kedua) berada pada koordinat yang sama persis.

Dengan sistem ini, realistis yang menjadi tujuan utama dari game tiga dimensi kurang dipenuhi karena mata manusia menganut sistem proyeksi yang kedua yaitu PERSPECTIVE PROJECTION.

1.2.2 Perspective Projection

Teknik inilah yang dipakai oleh game-game developer dalam membuat game tiga dimensi mulai dari flight simulator seperti F-15, Wing Commander, raytracer engine seperti DOOM, QUAKE dan lain sebagainya.

Secara geometris teknik ini memproyeksikan koordinat tiga dimensi objek ke koordinat dua dimensi layar monitor dengan cara menarik garis lurus dari objek ke mata kita. Koordinat X dan Y hasil proyeksi ialah koordinat X dan Y di mana garis tersebut menembus layar monitor.



Teknik Perspective Projection secara geometris.

Perhatikan garis lurus yang menghubungkan objek pada dunia tiga dimensi dengan mata kita. Garis ini menembus layar monitor pada titik P. Koordinat tiga dimensi dari objek ialah panjang garis BC sebagai koordinat Y, panjang garis DC sebagai koordinat X dan panjang garis DO sebagai koordinat Z. Di lain pihak, pada koordinat layar, koordinat X dari hasil proyeksi ialah panjang garis OQ sedangkan koordinat Y ialah panjang garis PQ.

Untuk mencari OQ dan PQ secara matematis, perhatikan segitiga AOQ dan segitiga ADC.

Dengan dalil segitiga kita dapat menyimpulkan bahwa:

OQ/DC = AO/AD

OQ = (AO x DC)/AD

OQ ialah koordinat X hasil proyeksi dari objek.
AO ialah jarak antara mata kita dengan monitor, biasanya pada mode resolusi 320 x 200 dipergunakan jarak standard = 400 pixel.
DC ialah koordinat X tiga dimensi dari objek.
AD ialah koordinat Z tiga dimensi dari objek.

Jadi persamaan OQ di atas dapat kita asumsikan sebagai:

Xs = (400 * X)/Z

Demikian pula untuk mencari PQ, perhatikan segitiga APQ dan segitiga ABC.

PQ/BC = AO/AD

PQ = (AO * BC)/AD

PQ ialah koordinat Y hasil proyeksi dari objek.
AO ialah jarak antara mata kita dengan monitor, biasanya pada mode resolusi 320 x 200 dipergunakan jarak standard = 400 pixel.
BC ialah koordinat Y tiga dimensi dari objek.
AD ialah koordinat Z tiga dimensi dari objek.

Jadi persamaan PQ di atas dapat kita asumsikan sebagai:

Ys = (400 * Y) / Z

Coba kita praktekan apa yang telah kita pelajari pada program di bawah ini.

```
{-------------------------------------------      }
{ Program     : B3PSV1.PAS                       }
{ Coder       : Agustinus Nalwan                 }
{ Description : Program proyeksi Titik dengan    }
{               Perspective Projection.          }
{-------------------------------------------      }

Uses Crt,Dos,VgaCad,VgaFont;

Const View_Distance = 400;

Var X,Y,Z:Real;
    Buff :Pointer;

Procedure Small;External;
{$L Small.Obj}

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Initialize;
  Begin
    OpenVga256;
    LoadFont(@Small);
    GetMem(Buff,64000);
    X:=0;Y:=0;Z:=600;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Function Strig(Num:Longint):String;
Var St:String;
  Begin
    Str(Num,St);
```

```
    Strig:=St;
  End;

Procedure Main;
Var Ch              :Char;
    pX,pY           :Longint;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        Psv_Projection(X,Y,Z,pX,pY);
        PutPixel(Buff^,pX,pY,13);
        PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
        PutString(Buff^,60,10,0,15,'Y : '+Strig(Round(Y)));
        PutString(Buff^,110,10,0,15,'Z : '+Strig(Round(Z)));
        MoveTo(Buff^,Screen^);
        Case Ch of
          #72:Y:=Y+2;
          #80:Y:=Y-2;
          #75:X:=X-2;
          #77:X:=X+2;
          '+','=':Z:=Z+2;
          '-','_':Z:=Z-2;
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Cobalah jalankan program di atas. Gerakkan koordinat Z dengan menggunakan tombol + dan -. WOW!!! Inilah hasil proyeksi yang kita inginkan. Pada saat koordinat Z berubah, koordinat dua dimensi pun ikut berubah.

PENJELASAN LISTING PROGRAM

Program ini memiliki format yang sama persis dengan program kita B3PRL1.PAS. Hanya di sini kita tidak menggunakan PARALLEL PROJECTION tetapi menggunakan PERSPECTIVE PROJECTION.

Bagian yang diganti hanyalah pada pemanggilan procedure untuk memproyeksikan titik. Di sini kita tidak menggunakan PLL_Projection tetapi menggunakan PSV_Projection.

```
Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
```

Seperti yang telah kita pelajari, bahwa proyeksi untuk X ialah View_Distance*X/Z di mana View_Distance di sini ialah konstanta berharga 400. Ditambahkan 160 untuk menggeser koordinat system pada tengah layar.

```
pX:=160+Round((View_Distance*X)/Z);
```

Dilakukan proses 100-Hasil proyeksi, dengan alasan yang sama yaitu terbaliknya sumbu Y pada koordinat tiga dimensi dengan sumbu Y pada koordinat dua dimensi.

```
    pY:=100-Round((View_Distance*Y)/Z);
  End;
```

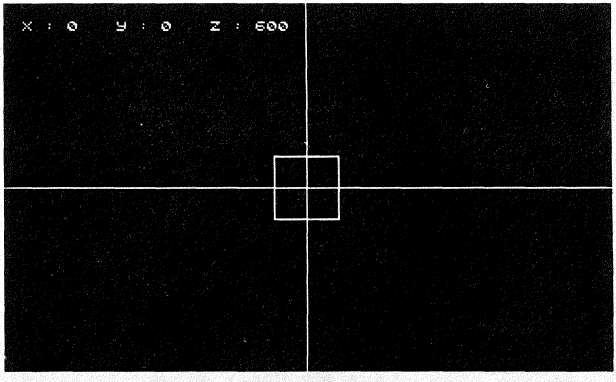
Sekarang kita akan mencoba membuat sebuah program seperti yang pernah kita buat, yaitu memproyeksikan persegi panjang tetapi dengan menggunakan PERSPECTIVE PROJECTION. Di sini akan terlihat dengan jelas tingkat realistis dari PERSPECTIVE PROJECTION yang telah kita pelajari.

```
{-----------------------------------------------     }
{ Program     : B3PSV2.PAS                          }
{ Coder       : Agustinus Nalwan                    }
{ Description : Program proyeksi Balok dengan       }
{               Perspective Projection.             }
{-----------------------------------------------     }

Uses Crt,Dos,VgaCad,VgaFont;

Const View_Distance = 400;

Var X,Y,Z:Real;
    Buff :Pointer;
    Width,Length:Longint;


Procedure Small;External;
{$L Small.Obj}

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;


Procedure Initialize;
  Begin
    OpenVga256;
    LoadFont(@Small);
    GetMem(Buff,64000);
    X:=0;Y:=0;Z:=600;
    Length:=50;Width:=50;
  End;


Procedure DeInitialize;
  Begin
```

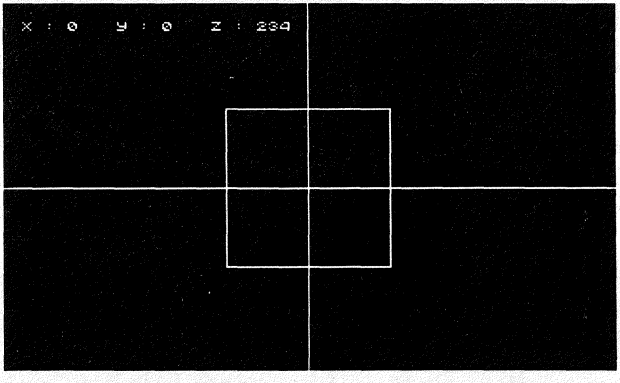
```
    CloseVga256;
  End;

Function Strig(Num:Longint):String;
Var St:String;
  Begin
    Str(Num,St);
    Strig:=St;
  End;

Procedure Main;
Var Ch            :Char;
    pX1,pY1,pX2,pY2:Longint;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        Psv_Projection(X-Length div 2,Y+Width div 2,Z,pX1,pY1);
        Psv_Projection(X+Length div 2,Y-Width div 2,Z,pX2,pY2);
        Rectangle(Buff^,pX1,pY1,pX2,pY2,15);
        PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
        PutString(Buff^,60,10,0,15,'Y : '+Strig(Round(Y)));
        PutString(Buff^,110,10,0,15,'Z : '+Strig(Round(Z)));
        MoveTo(Buff^,Screen^);
        Case Ch of
          #72:Y:=Y+2;
          #80:Y:=Y-2;
          #75:X:=X-2;
          #77:X:=X+2;
          '+','=':Z:=Z+2;
          '-','_':Z:=Z-2;
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Sekarang coba gerakkan koordinat Z dengan menggunakan tombol - dan +. Terlihat bahwa, persegi panjang ikut mengecil saat koordinat Z kita tambah dan membesar jika koordinat Z kita kurangi.



Hasil proyeksi dari kotak pada koordinat Z = 600.



Hasil proyeksi dari kotak pada koordinat Z = 234 terlihat lebih besar.

PENJELASAN LISTING PROGRAM

Pada main program, dipanggil fungsi Perspective Projection dua kali untuk memproyeksikan titik kiri atas dan kanan bawah dari persegi panjang.

```
Procedure Main;
Var Ch              :Char;
    pX1,pY1,pX2,pY2:Longint;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        Psv_Projection(X-Length div 2,Y+Width div 2,Z,pX1,pY1);
        Psv_Projection(X+Length div 2,Y-Width div 2,Z,pX2,pY2);
        Rectangle(Buff^,pX1,pY1,pX2,pY2,15);
        PutString(Buff^,10,10,0,15,'X : '+Strig(Round(X)));
```

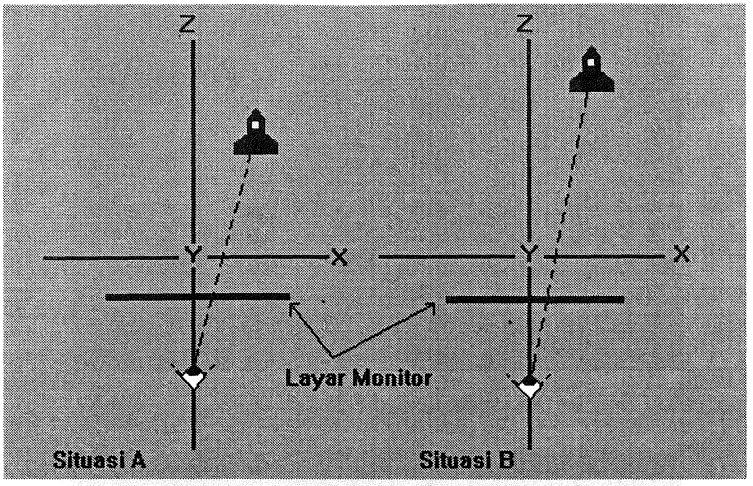
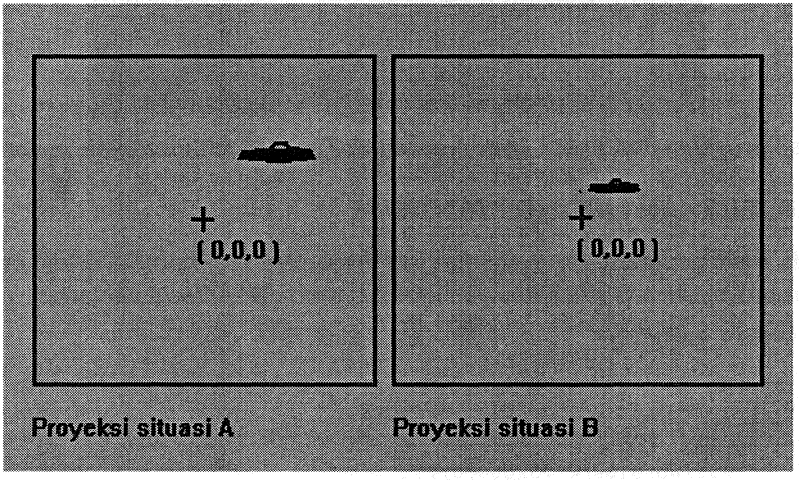


Diagram di atas menunjukkan sebuah objek pada dua situasi yang berbeda.

Nah, sekarang dengan situasi A dan B di atas (di mana pada situasi B, koordinat Z dari objek lebih jauh sedangkan X dan Y tetap sama) kita coba memproyeksikannya dengan menggunakan teknik perspective projection.



Hasil proyeksi dengan menggunakan sistem perspective projection.

Karena sistem perspective projection tidak mengabaikan koordinat Z dari objek, bahkan menggunakan koordinat Z sebagai pembagi dari koordinat tiga dimensi X dan Y dari objek, maka semakin jauh/besar koordinat Z-nya maka semakin dekat hasil proyeksi tersebut dengan titik origin (0,0,0). Perhatikan pada situasi B, objek lebih dekat ke titik origin.

Karena di sini kita melakukan proyeksi secara perspective, maka view volume (daerah tiga dimensi yang terlihat di layar monitor) akan berupa piramid, sedangkan pada parallel projection, view volume akan berupa balok.

Amatlah penting untuk benar-benar memahami apa yang telah dijelaskan pada bab ini, karena ini adalah dasar dari pemrograman tiga dimensi. Bila Anda masih mengalami kesulitan dalam memahaminya, bacalah berulang-ulang dan teruslah bereksperimen dengan memodifikasi program contoh yang telah diberikan.

Sebenarnya masih ada beberapa teknik proyeksi tiga dimensi yang lain tetapi tidak akan kita bahas karena di samping tingkat realistisnya tidak memadai, juga kita tidak akan pernah menggunakannya.

1.3 Dasar-Dasar Geometri dan Transformasi Tiga Dimensi

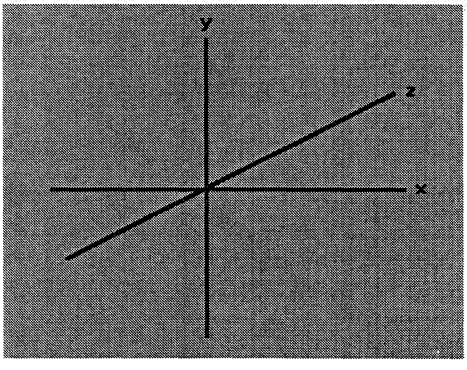
Pada bagian ini kita akan mempelajari dasar-dasar transformasi dan istilah-istilah dalam dunia geometri tiga dimensi.

1.3.1 Coordinate System

AXIS ASSIGNMENT

Axis assignment ialah pemberian label terhadap ketiga sumbu koordinat yaitu sumbu horizontal, sumbu vertikal dan sumbu kedalaman. Sebenarnya axis assignment ini bebas, kita dapat menganggap sumbu horizontal sebagai Z, vertikal sebagai X dan kedalaman sebagai Y. Untuk lebih memudahkan kita, lebih baik kita buat axis assigment standard yang akan selalu kita pakai sebagai pedoman dalam buku ini. Ok?

Axis assignment standard kita ialah, sumbu X sebagai sumbu horizontal, sumbu Y sebagai sumbu vertikal, sumbu Z sebagai sumbu kedalaman.



AXIS POLARITY ASSIGNMENT

Pada koordinat dua dimensi biasa, di mana kita mengenal sumbu X dan Y, tentunya Anda ingat bahwa sumbu X yang terletak di belahan kiri dari titik origin ialah sumbu X negatif sedangkan pada bagian kanan ialah sumbu X positif, demikian pula pada sumbu Y. Di koordinat tiga dimensi, kita juga mengenal axis polarity (positif/negatif). Masing-masing sumbunya yaitu X, Y dan Z memiliki bagian negatif dan positif juga. Sebagaimana halnya dengan axis assignment, dalam menentukan polaritas dari axis tidak ada ketentuan/bebas. Tetapi untuk mempermudah, akan kita tentukan polaritas standard.

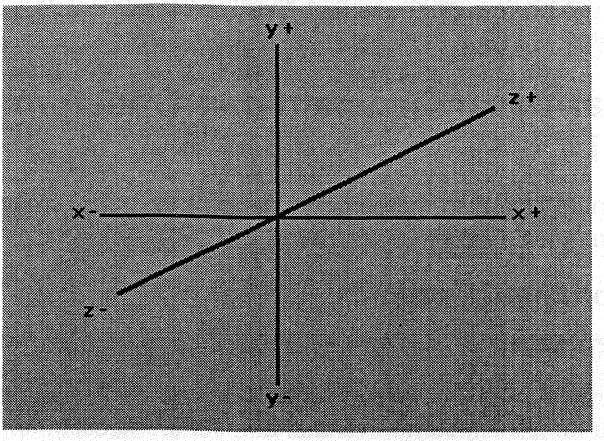
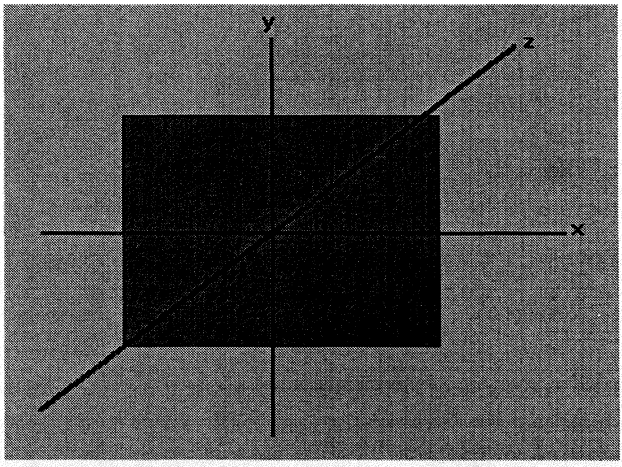


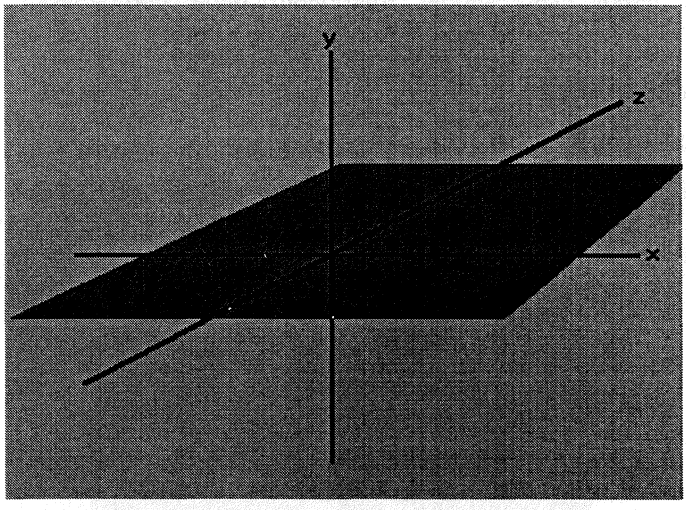
Diagram di atas menunjukkan axis polarity standard yang akan selalu kita pergunakan.

1.3.2 Coordinate Plane

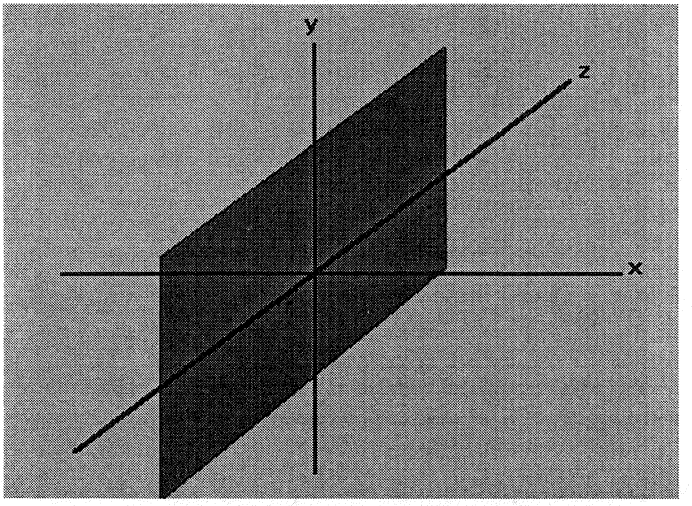
Dalam dunia tiga dimensi, kita memiliki tiga buah coordinate plane/bidang. Pertama ialah XY plane. Bidang ini ialah bidang yang tegak lurus dengan Z axis.



Bidang yang kedua ialah XZ plane seperti pada gambar di bawah ini. Bidang ini tegak lurus dengan sumbu Y.



Bidang yang terakhir ialah YZ plane yaitu bidang yang tegak lurus dengan sumbu X.

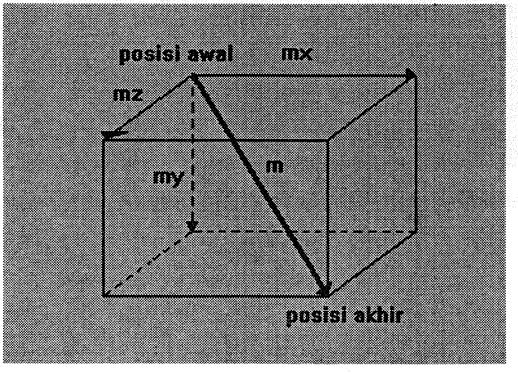


Ketiga bidang ini sebenarnya tidak banyak berarti dalam pemrograman tiga dimensi, tetapi akan sangat membantu pada pembahasan bab-bab selanjutnya.

1.3.3 Movement

DILATASI

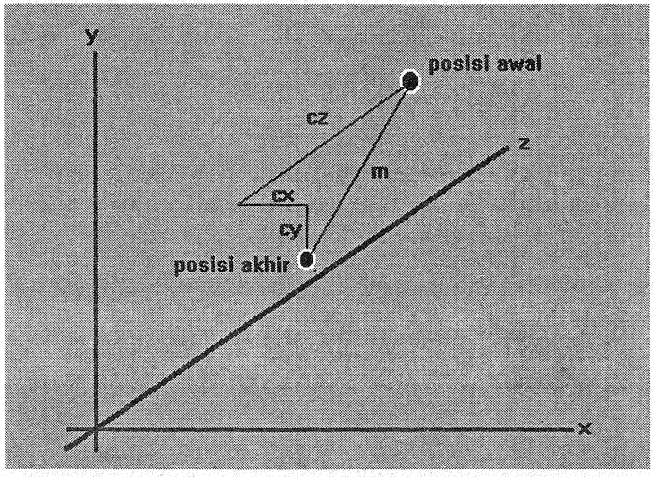
Dilatasi ialah pergeseran/perpindahan tempat sebuah objek dalam dunia tiga dimensi. Dilatasi pada koordinat tiga dimensi adalah sebuah vektor yang memiliki tiga komponen yaitu perpindahan sumbu X, sumbu Y dan sumbu Z.



Perhatikan diagram di atas. Garis yang berlabelkan m itu adalah vektor perpindahan dari sebuah titik. Vektor m ini memiliki tiga buah komponen yaitu mx, my dan mz yang dapat diproyeksikan terhadap sumbu koordinatnya masing-masing berupa sebuah garis. Dengan kata lain mx

ialah perpindahan koordinat X dari objek, my adalah perpindahan koordinat Y dari objek sedangkan mz adalah perpindahan koordinat Z dari objek.

Segala kemungkinan dilatasi dapat dinyatakan dalam mx, my dan mz sehingga akan memudahkan kita dalam menggerakkan objek pada dunia tiga dimensi.

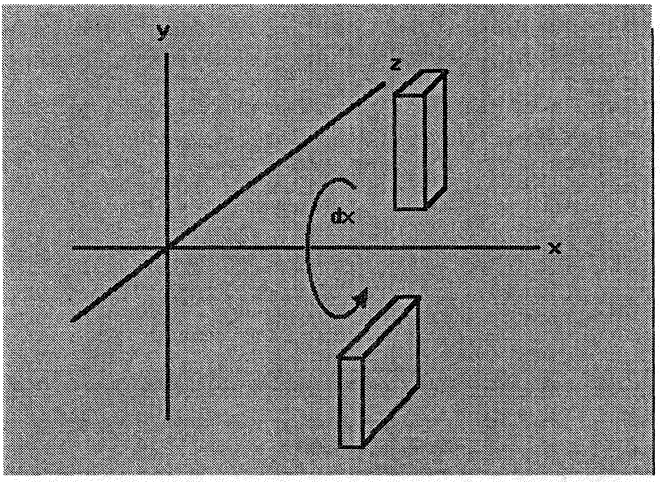


Pada diagram di atas ditunjukkan sebuah benda dari posisi awal berpindah sejauh vektor m menuju posisi akhir. Vektor m ini diproyeksikan menjadi tiga komponennya yaitu cx, cy dan cz. Mula-mula dilakukan penambahan pada koordinat Z sejauh cz kemudian dilanjutkan dengan penambahan koordinat X sejauh cx dan terakhir dilakukan penambahan koordinat Y sejauh cy. Urutan operasi dilatasi ini dapat diputarbalikkan/bersifat komutatif dan akan selalu mendapatkan hasil yang sama.

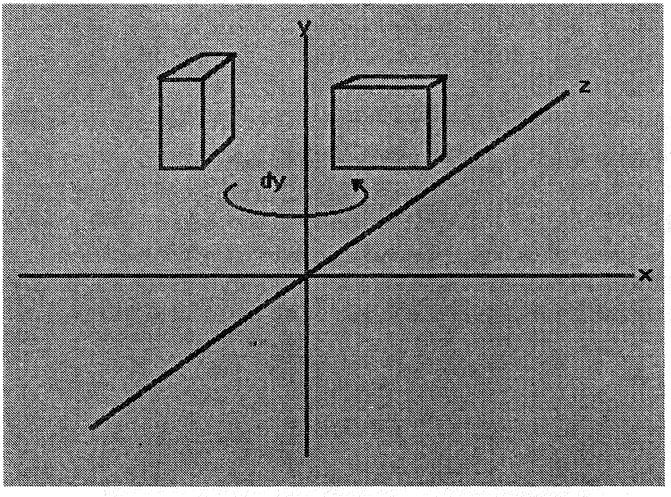
ROTASI

Jenis perpindahan yang kedua ialah rotasi. Rotasi berarti berputar. Dalam dunia tiga dimensi, kita mengenal tiga jenis rotasi yaitu rotasi terhadap sumbu X, rotasi terhadap sumbu Y dan rotasi terhadap sumbu Z.

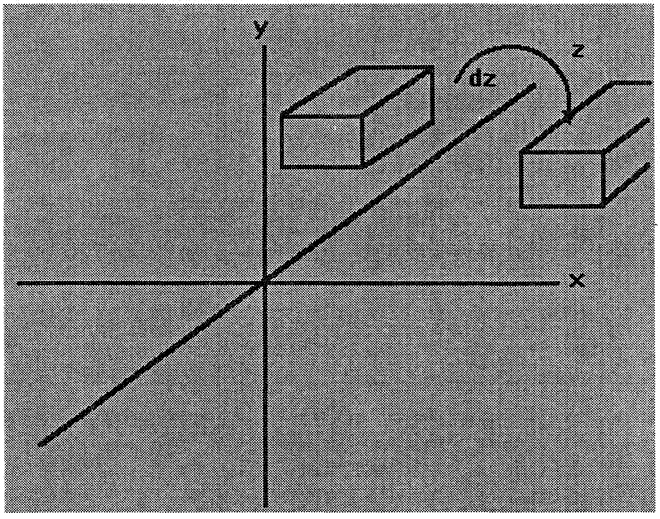
Rotasi sumbu X ialah rotasi yang dilakukan terhadap sumbu X. Kedengarannya sederhana bukan? Demikian pula dengan rotasi sumbu Y dan Z yang dilakukan terhadap sumbu Y dan sumbu Z.



Gambar di atas menunjukkan sebuah objek yang dirotasi terhadap sumbu X sejauh dx.



Ini adalah rotasi yang dilakukan terhadap sumbu Y sejauh dy.



Ini adalah rotasi yang dilakukan terhadap sumbu Z sejauh dz.

BAB 2
PROYEKSI OBJEK SECARA BITMAP

2.1 Pendahuluan

Pada bab ini kita akan membahas sedikit mengenai bitmap tiga dimensi tanpa disertai dengan prakteknya karena pembahasan kita pada edisi ini akan difokuskan pada pemrograman objek vektor.

Pada bab yang lalu, telah kita bahas bagaimana teknik proyeksi sebuah titik dari koordinat tiga dimensi ke koordinat layar dua dimensi. Pada game tiga dimensi, kita tidak cukup hanya mampu memproyeksikan sebuah titik yang berkeliaran di dunia tiga dimensi saja tetapi kita harus mampu memproyeksikan sebuah objek (pesawat, orang, pohon, mobil dan sebagainya).

Dalam hal ini dikenal dua buah teknik yang terus dikembangkan oleh para developer game dalam merakit game-game tiga dimensi profesional. Teknik yang pertama ialah BITMAP yang dapat di-resize. (Ingat PUT3D pada SAP PEMROGRAMAN ANIMASI DAN GAME PROFESIONAL 2.) Di sini kita menyediakan objek tersebut dalam bentuk sprite dua dimensi. Game yang termasuk pelopor teknik ini ialah game-game SEGA seperti Space Harrier, After Burner. Kedua game ini telah sukses di-port (dimodifikasi dan di-compile ulang pada machine platform yang berbeda) ke komputer.

Karena keterbatasan komputer pada saat itu yaitu masih menggunakan processor 286, maka teknik BITMAP resizing seperti yang telah kita pelajari pada edisi lalu diganti dengan menyediakan sprite objek tersebut dalam beberapa ukuran. Memang cara ini sangat mempercepat jalannya program karena teknik BITMAP resizing pada PUT3D membutuhkan cycle (jumlah

CPU instruction) sangat banyak tetapi memakan jumlah memori yang jauh lebih besar mengingat kita perlu menyediakan sprite dalam beberapa ukuran. Di samping memboroskan memori, cara ini juga mengakibatkan animasi yang dihasilkan kurang halus, karena jumlah ukuran yang tersedia tentu saja tidak mungkin dapat memenuhi segala kemungkinan ukuran dari objek yang dapat terjadi melainkan kita mengambil ukuran yang terdekat yang ada dari ukuran yang diinginkan.

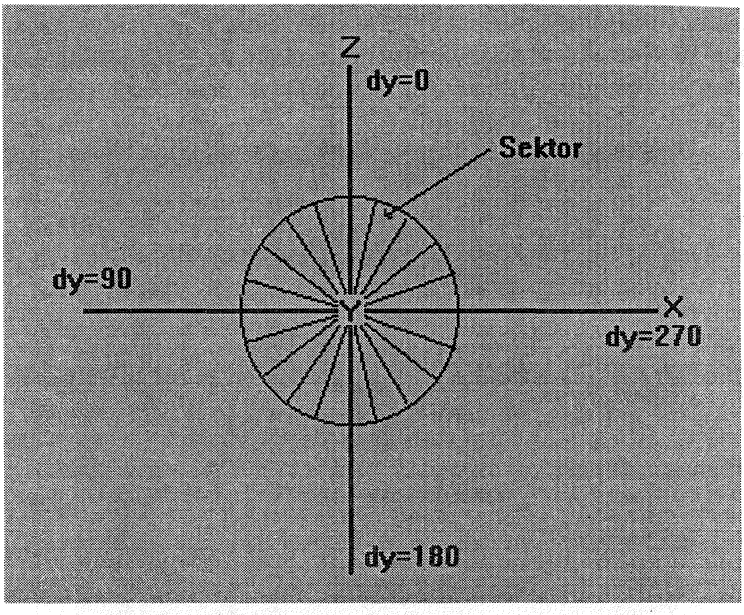
Untuk standar komputer 386 ke atas, teknik ini tidak lagi digunakan karena bitmap resizing telah memiliki speed yang memungkinkan untuk diterapkan pada game.

2.2 Sprite Bank Rotation Technique

Nah langkah selanjutnya ialah bagaimana membuat sebuah objek yang dapat berputar, dengan kata lain kita dapat melihat sisi kanan, kiri, belakang dari objek tersebut. Karena kita menggunakan teknik bitmap maka satu-satunya cara untuk menciptakan efek ini ialah dengan menyediakan beberapa gambar dari objek yang dilihat dari sudut pandang yang berbeda-beda (SPRITE BANK ROTATION TECHNIQUE).

Tingkat kehalusan animasi yang dihasilkan akan tergantung pada jumlah gambar objek dari sudut pandang yang berbeda-beda. Semakin banyak jumlah gambar yang disediakan, semakin halus animasi yang akan dihasilkan.

Misalkan kita telah menentukan jumlah gambar yang akan disediakan ialah 20. Maka kita harus menyediakan gambar objek dari 20 arah yang berbeda. Jadi tiap gambar akan mewakili sudut sebesar 360/20 = 18 derajat. Dengan kata lain, jika objek dilihat dari sudut 0 derajat hingga 17 derajat, maka tampilkan sprite pertama, jika objek dilihat dari sudut 18 derajat hingga 35 derajat, tampilkan sprite kedua, demikian seterusnya.



Pembagian sektor dalam pemilihan sprite yang harus ditampilkan dilihat dari sudut pandang atas.

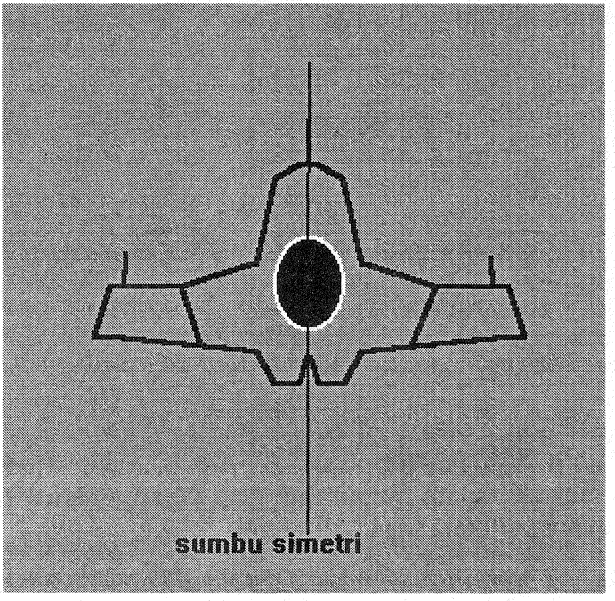
Pada gambar di atas divisualisasikan pembagian sektor dengan jumlah gambar yang disediakan sebanyak 20. Sehingga masing-masing sektor mewakili sudut sebesar 18 derajat.

Seperti yang telah dibahas sebelumnya, jika pembagian sektor lebih banyak, jumlah gambar yang diperlukan untuk mewakili masing-masing sektor juga harus lebih banyak dan besar sudut yang diwakili masing-masing sektornya menjadi lebih kecil sehingga animasi akan terlihat lebih halus.

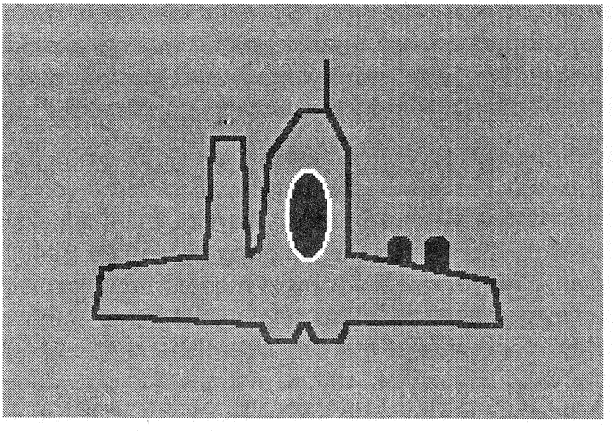
Memang game-game yang memanfaatkan teknik ini perlu mengorbankan memori yang cukup besar sebagai penyimpanan sprite yang diambil dari berbagai posisi. Coba bayangkan saja, jika kita menggunakan 20 sektor sedangkan besar memori untuk menampung satu sprite ialah 10 kilobytes, maka diperlukan 20 x 10 = 200 kilobytes memori untuk sebuah objek. Hal ini tentu saja merupakan ancaman bagi game-game kuno yang hanya mempergunakan conventional memori sejumlah 640 kbytes. Sebenarnya ada satu cara yang digunakan untuk menghemat penggunaan memori dalam penggunaan SPRITE BANK ROTATION TECHNIQUE dan ternyata cara ini memang benar-benar dipergunakan pada game-game seperti Wolfenstein 3D bahkan DOOM, HERETIC, HEXEN sekalipun.

Karena kebanyakan objek yang ditampilkan bersifat simetris (sisi kiri sama dengan sisi kanan) maka cukup dilakukan penyimpanan sprite separuh dari

total sektor yang dipergunakan. Yaitu kita sediakan sprite hanya pada sektor 1 hingga 10. Sedangkan sisanya dapat dilakukan dengan cara menampilkan sprite nomor 1 hingga 10 secara terbalik (kiri jadi kanan, kanan jadi kiri).



Objek yang simetris.



Objek yang tidak simetris.

Dengan ini kita telah menghemat separuh jumlah memori yang kita pergunakan sebelumnya.

2.3 Pengembangan Sprite Bank Rotation Technique

Sprite Bank Rotation Technique ini hanya mampu memproyeksikan objek yang dapat berputar hanya pada sumbu Y. Sehingga kita mampu melihat sisi kanan, kiri, belakang maupun depan dari objek. Tetapi, bayangkan kalau kita membuat sebuah game flight simulator seperti Wing Commander. Kita tidak hanya mampu melihat sisi kiri, kanan, belakang, depan dari pesawat musuh. Kita dapat melihat sisi atas maupun bawah dari pesawat musuh. WOW!!! Bagaimana cara ORIGIN menciptakan efek ini?

Sebenarnya secara teori efek ini dilakukan dengan mengembangkan Sprite Bank Rotation Technique dengan cara selain membagi sektor pada bidang XZ, dilakukan pula pembagian sektor pada bidang YZ untuk masing-masing sektor bidang XZ-nya. Sehingga masing masing sektor pada bidang XZ masih dibagi menjadi beberapa sektor bidang YZ.

Tentu saja dengan cara ini terjadi pembengkakan jumlah memori yang diperlukan karena jumlah sektor yang ada akan menjadi 20 x 20 = 400 sektor. Dan dengan cara pemanfaatan simetri maka jumlah sprite yang harus disimpan ialah 10 x 10 = 100 sektor.

Setelah saya selidiki lebih lanjut, pada Wing Commander digunakan 12 sektor pada bidang XZ dan 12 sektor pada bidang YZ sehingga jumlah sprite yang harus disimpan ialah 6 x 6 = 36. Ukuran sprite pesawat tempur normal pada Wing Commander ialah sebesar 60 x 40 yang akan memakan memori sebesar 2400 bytes. Sehingga jumlah memori yang diperlukan untuk menyimpan sebuah objek ialah 2400 x 36 = 86 kbytes. Dengan teknik kompresi yang telah kita pelajari untuk menyimpan masing-masing sprite-nya, jumlah 86 kbytes ini bisa berkurang menjadi sekitar 64 kbytes. WOW!!! Kita baru saja memecahkan apa yang kira-kira dilakukan ORIGIN dalam menciptakan Wing Commander.

Demikianlah pembahasan kita mengenai bitmap tiga dimensi. Pada bab-bab selanjutnya kita akan mulai membahas programming tiga dimensi yang sebenarnya.

BAB 3

PROYEKSI OBJEK SECARA VEKTOR

3.1 3D Vector Objek Data Structure

Teknik proyeksi yang kedua ialah teknik VEKTOR (Vector). Di sini objek dibentuk dari poligon-poligon yang menghubungkan sejumlah titik. Teknik ini dipelopori oleh sebuah game yang amat kuno (sekitar tahun 1980). Processor pada saat itu masih menggunakan XT dan 286.

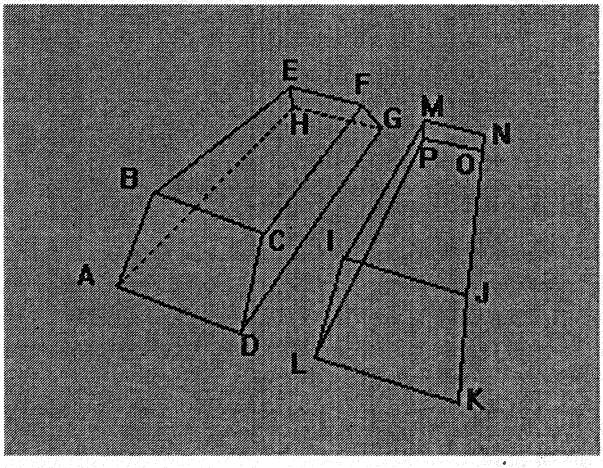


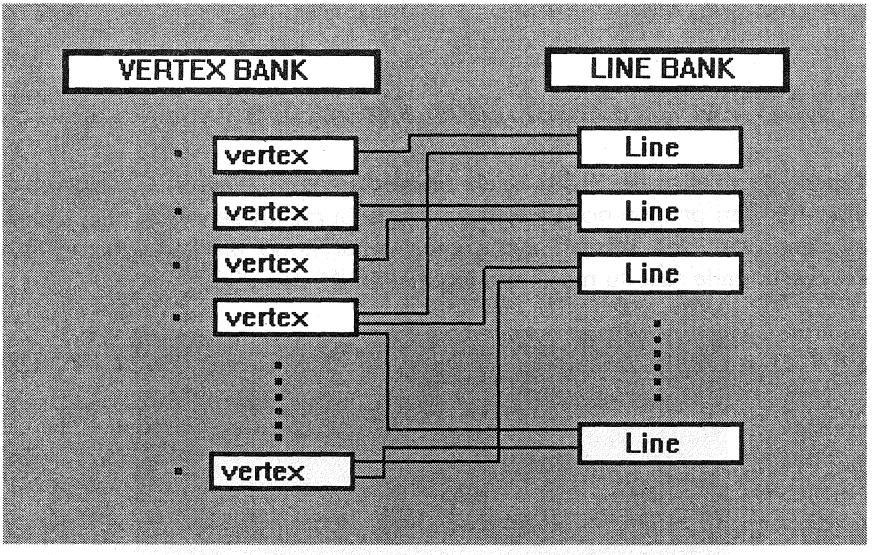
Diagram di atas menunjukkan sebuah objek yang dibentuk dari sejumlah titik (A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P) yang dihubungkan oleh garis-garis.

Inilah teknik vektor yang pertama kalinya diterapkan dalam pembuatan game. Teknik ini dinamakan wireframe projection.

Memang hasilnya kurang memuaskan karena sebuah objek terlihat hanya berupa garis-garis dan tembus pandang jika dibandingkan dengan teknik BITMAP resizing.

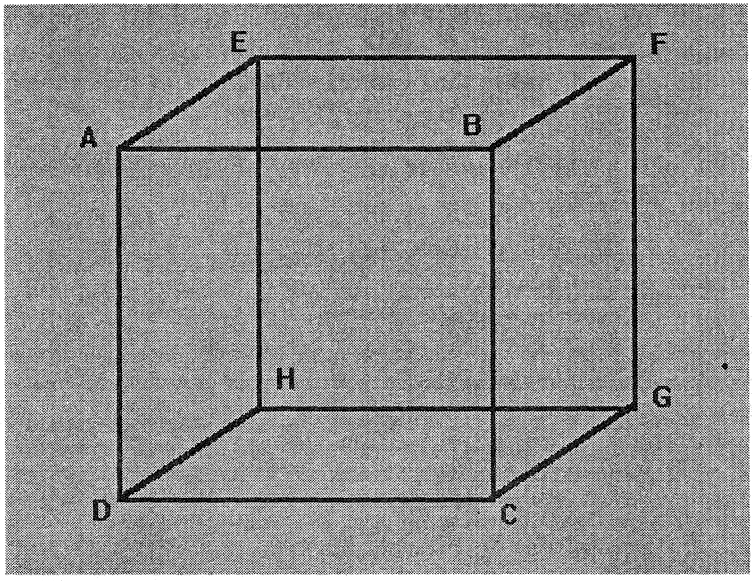
Wireframe projection ini dilakukan dengan cara mendefinisikan sebuah objek sebagai array yang berisi koordinat masing-masing titik yang membentuk objek tersebut disertai dengan data hubungan masing-masing titik dengan titik yang lainnya. Kemudian pada saat projection dilakukan, masing-masing titik akan diproyeksikan ke koordinat dua dimensi. Proses terakhir ialah menampilkan garis-garis yang menghubungkan titik-titik hasil proyeksi pada layar monitor. Nggak sulit kan?

Untuk lebih jelasnya perhatikan gambar di bawah ini.

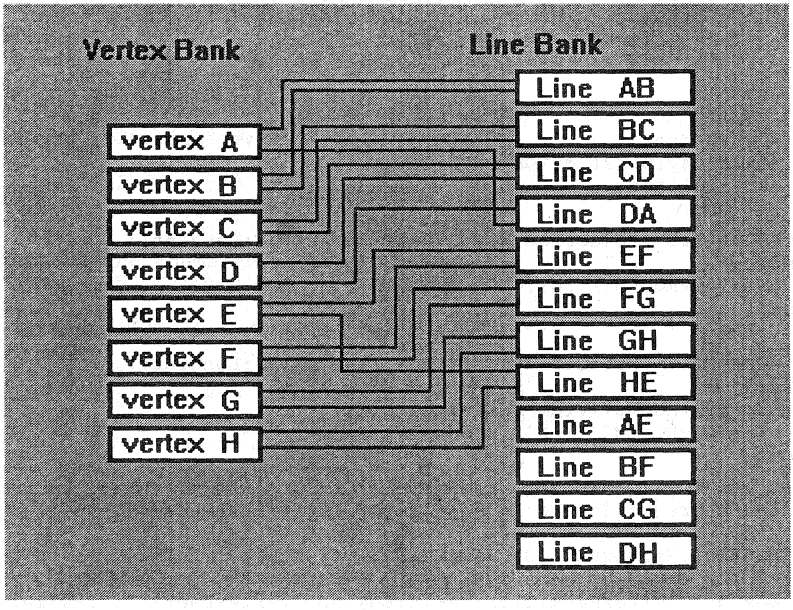


Untuk sebuah objek, kita memiliki dua buah bank yaitu Vertex Bank yang berisi titik-titik sudut dari objek. Juga Line Bank yang berisi dua index titik pada Vertex Bank yang hendak dihubungkan. Pada setiap Vertex di dalam Vertex Bank memiliki koordinat X, Y dan Z sebagai informasi koordinat dari titik/vertex tersebut.

Di bawah ini adalah sebuah objek yang akan disimpan dalam format yang telah dijelaskan di atas.



Perhatikan, jumlah titik pada balok ini ada 8 yaitu A,B,C,D,E,F,G,H. Sedangkan jumlah garis pada balok ini ialah 12 yaitu AB, BC, CD, DA, EF, FG, GH, HE, AE, BF, CG, DH. Untuk itu diperlukan Vertex Bank sebesar 8 cell dan Line Bank sebesar 12 cell.



Inilah bentuk penyimpanan objek balok tersebut ke dalam format Vertex Bank dan Line Bank. Teknik ini sering dipergunakan dalam dunia pemrograman tiga dimensi. Salah satu contoh software yang mempergunakan teknik ini ialah 3D Studio pada format file 3DS-nya.

Perhatikan bahwa tiap garis dalam Line Bank menunjukkan kedua buah vertex pada Vertex Bank.

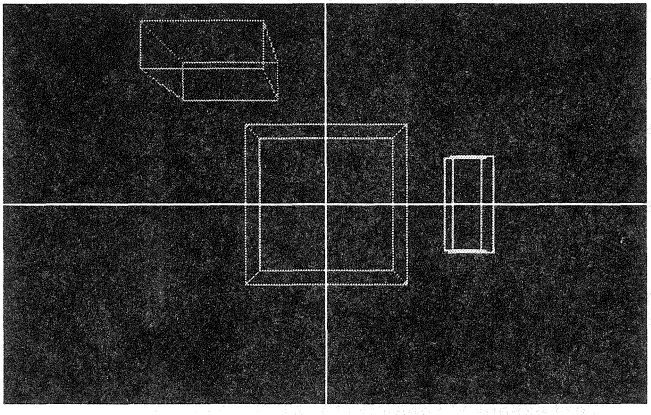
```
{-------------------------------------------------------      }
{ Program     : B3VEC1.PAS                                    }
{ Coder       : Agustinus Nalwan                              }
{ Description: Merancang Objek dengan teknik vektor           }
{-------------------------------------------------------      }
Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Line_Rec = Record
       A,B : Integer;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
     End;

     ObjRect = Record
       Color      : Byte;
       LineBank   : Array[0..20] of Line_Rec;
       VertexBank : Array[0..20] of Vertex_Rec;
       TotalVertex: Integer;
       TotalLine  : Integer;
     End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=x-lx;
       Box[No].VertexBank[0].Y:=y+ly;Box[No].VertexBank[0].Z:=z-lz;
    Box[No].VertexBank[1].X:=x+lx;Box[No].VertexBank[1].Y:=y+ly;
       Box[No].VertexBank[1].Z:=z-lz;
```

```
    Box[No].VertexBank[2].X:=x+ lx;Box[No].VertexBank[2].Y:=y-ly;
        Box[No].VertexBank[2].Z:=z-lz;
    Box[No].VertexBank[3].X:=x-lx;Box[No].VertexBank[3].Y:=y-ly;
        Box[No].VertexBank[3].Z:=z-lz;
    Box[No].VertexBank[4].X:=x-lx;
        Box[No].VertexBank[4].Y:=y+ly;Box[No].VertexBank[4].Z:=z+lz;
    Box[No].VertexBank[5].X:=x+lx;Box[No].VertexBank[5].Y:=y+ly;
        Box[No].VertexBank[5].Z:=z+lz;
    Box[No].VertexBank[6].X:=x+lx;Box[No].VertexBank[6].Y:=y-ly;
        Box[No].VertexBank[6].Z:=z+lz;
    Box[No].VertexBank[7].X:=x-lx;Box[No].VertexBank[7].Y:=y-ly;
        Box[No].VertexBank[7].Z:=z+lz;
    Box[No].LineBank[0].A:=0; Box[No].LineBank[0].B:=1;
    Box[No].LineBank[1].A:=1; Box[No].LineBank[1].B:=2;
    Box[No].LineBank[2].A:=2; Box[No].LineBank[2].B:=3;
    Box[No].LineBank[3].A:=3; Box[No].LineBank[3].B:=0;
    Box[No].LineBank[4].A:=4; Box[No].LineBank[4].B:=5;
    Box[No].LineBank[5].A:=5; Box[No].LineBank[5].B:=6;
    Box[No].LineBank[6].A:=6; Box[No].LineBank[6].B:=7;
    Box[No].LineBank[7].A:=7; Box[No].LineBank[7].B:=4;
    Box[No].LineBank[8].A:=0; Box[No].LineBank[8].B:=4;
    Box[No].LineBank[9].A:=1; Box[No].LineBank[9].B:=5;
    Box[No].LineBank[10].A:=2; Box[No].LineBank[10].B:=6;
    Box[No].LineBank[11].A:=3; Box[No].LineBank[11].B:=7;
    Box[No].TotalVertex:=8;
    Box[No].TotalLine:=12;
    Box[No].Color:=Col;
  End;

Procedure Initialize;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure Wire_Frame_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalLine-1 do
          Begin
           Psv_Projection(VertexBank[LineBank[A].A].X,
            VertexBank[LineBank[A].A].Y,
             VertexBank[LineBank[A].A].Z,X1,Y1);
           Psv_Projection(VertexBank[LineBank[A].B].X,
            VertexBank[LineBank[A].B].Y,
             VertexBank[LineBank[A].B].Z,X2,Y2);
           Line(Buff^,X1,Y1,X2,Y2,Color);
          End;
      End;
  End;
```

```
Procedure Display_Objek(No:Integer);
  Begin
    Wire_Frame_Projection(No);
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Display_Objek(0);
        Display_Objek(1);
        Display_Objek(2);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

WOW!!! Kita telah berhasil memproyeksikan tiga buah objek yang masing-masing berupa balok.



Tiga buah balok yang diproyeksikan secara wireframe.

PENJELASAN LISTING PROGRAM

```
Uses Crt,Dos,VgaCad;
```

Tentukan jarak antara mata kita dengan monitor yaitu 400 pixel.

```
Const View_Distance = 400;
Type
```

Ini adalah struktur data sebuah element dari Line Bank. Untuk menunjuk kedua buah titik pada Vertex Bank, digunakan dua variabel integer sebagai index yaitu A dan B.

```
  Line_Rec = Record
    A,B : Integer;
  End;
```

Pada sebuah Vertex Bank sendiri terdapat koordinat X, Y dan Z yang bertipe Real. Ketiga variabel ini digunakan untuk menyimpan koordinat tiga dimensi dari titik.

```
  Vertex_Rec = Record
    X, Y, Z : Real;
  End;
```

Sebuah objek memiliki data structure sebagai berikut:

```
  ObjRect = Record
```

Warna dari objek.

```
    Color      : Byte;
```

Line Bank dan Vertex Bank. Kita asumsikan jumlah Line tidak lebih dari 21 dan jumlah Vertex tidak lebih dari 21.

```
    LineBank   : Array[0..20] of Line_Rec;
    VertexBank : Array[0..20] of Vertex_Rec;
```

Dua variabel integer yang menunjukkan jumlah Line dan Vertex dari objek.

```
    TotalVertex: Integer;
    TotalLine  : Integer;
  End;

Var A,B:Integer;
```

Deklarasikan array untuk menyimpan tiga buah objek.

```
  Box:array[0..2] of ObjRect;
```

Variabel untuk keperluan Virtual Screen.

```
  buff:Pointer;
```

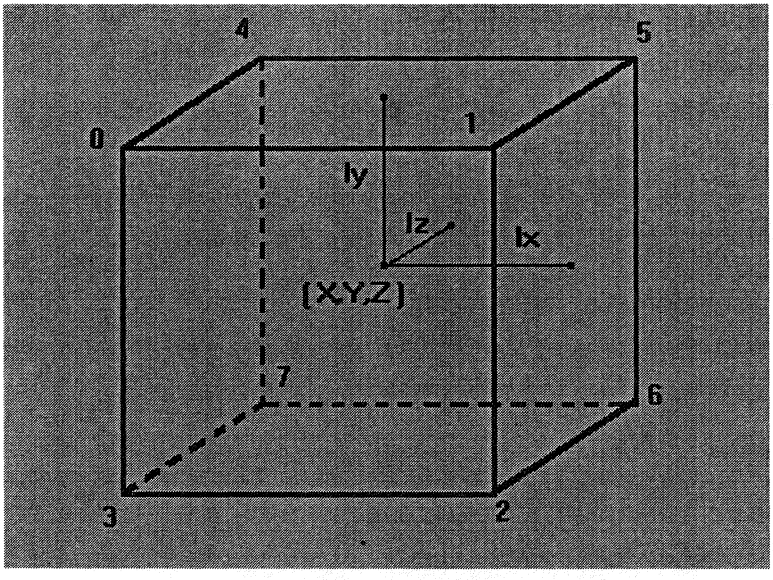
Di sini kita masukkan dua buah procedure PROJECTION yang telah kita

pelajari sehingga sewaktu-waktu kita dapat menukarnya untuk membandingkan hasilnya.

```
Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;
```

Procedure di bawah ini berfungsi untuk menciptakan sebuah balok yang disusun dalam data structure. Adapun parameter dari procedure ini ialah:

No	yaitu nomor objek yang hendak dirancang.
X,Y,Z	yaitu koordinat titik pusat dari balok.
lx,ly,lz	yaitu setengah panjang, setengah lebar dan setengah kedalaman dari balok.




```
Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=x-lx;Box[No].VertexBank[0].Y:=y+ly;
        Box[No].VertexBank[0].Z:=z-lz;
    Box[No].VertexBank[1].X:=x+lx;Box[No].VertexBank[1].Y:=y+ly;
```

```
    Box[No].VertexBank[1].Z:=z-lz;
  Box[No].VertexBank[2].X:=x+ lx;Box[No].VertexBank[2].Y:=y-ly;
    Box[No].VertexBank[2].Z:=z-lz;
  Box[No].VertexBank[3].X:=x-lx;Box[No].VertexBank[3].Y:=y-ly;
    Box[No].VertexBank[3].Z:=z-lz;
  Box[No].VertexBank[4].X:=x-lx;Box[No].VertexBank[4].Y:=y+ly;
    Box[No].VertexBank[4].Z:=z+lz;
  Box[No].VertexBank[5].X:=x+lx;Box[No].VertexBank[5].Y:=y+ly;
    Box[No].VertexBank[5].Z:=z+lz;
  Box[No].VertexBank[6].X:=x+lx;Box[No].VertexBank[6].Y:=y-ly;
    Box[No].VertexBank[6].Z:=z+lz;
  Box[No].VertexBank[7].X:=x-lx;Box[No].VertexBank[7].Y:=y-ly;
    Box[No].VertexBank[7].Z:=z+lz;
```

Isi LineBank dengan nomor index dari titik VertexBank yang dipakai. Sebagai contoh, pada baris pertama di bawah ini dapat diartikan sebagai garis nomor 0 menghubungkan titik pada Vertex Bank nomor 0 dengan Vertex Bank nomor 1. Perhatikan gambar di atas.

```
  Box[No].LineBank[0].A:=0; Box[No].LineBank[0].B:=1;
  Box[No].LineBank[1].A:=1; Box[No].LineBank[1].B:=2;
  Box[No].LineBank[2].A:=2; Box[No].LineBank[2].B:=3;
  Box[No].LineBank[3].A:=3; Box[No].LineBank[3].B:=0;
  Box[No].LineBank[4].A:=4; Box[No].LineBank[4].B:=5;
  Box[No].LineBank[5].A:=5; Box[No].LineBank[5].B:=6;
  Box[No].LineBank[6].A:=6; Box[No].LineBank[6].B:=7;
  Box[No].LineBank[7].A:=7; Box[No].LineBank[7].B:=4;
  Box[No].LineBank[8].A:=0; Box[No].LineBank[8].B:=4;
  Box[No].LineBank[9].A:=1; Box[No].LineBank[9].B:=5;
  Box[No].LineBank[10].A:=2; Box[No].LineBank[10].B:=6;
  Box[No].LineBank[11].A:=3; Box[No].LineBank[11].B:=7;
```

Setelah seluruh garis dihubungkan ke masing-masing titik penghubungnya, sekarang diisikan jumlah garis dan vertex yang ada pada objek, juga tentukan warna dari objek.

```
    Box[No].TotalVertex:=8;
    Box[No].TotalLine:=12;
    Box[No].Color:=Col;
  End;
```

Instalasi mode grafik, pesan memori untuk virtual screen.

```
Procedure Initialize;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
```

Ciptakan tiga buah balok pada dunia tiga dimensi kita.

Balok pertama berkoordinat, nomor objek = 0, memiliki koordinat (0,0,550) dan berukuran panjang = 100, lebar = 100 dan kedalaman = 100 (mengingat parameter yang dikirimkan adalah setengah dari panjang, lebar dan kedalaman yang akan diciptakan). Warna balok pertama ialah nomor warna 13.

Demikian seterusnya dengan balok kedua dan ketiga.

```
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
  End;
```

Kembali ke mode text.

```
Procedure DeInitialize;
  Begin
    CloseVga256;
  End;
```

Procedure di bawah ini berguna untuk memproyeksikan objek nomor NO dalam bentuk wire frame projection.

```
Procedure Wire_Frame_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
```

Lakukan loop untuk mengakses seluruh garis pada Line Bank.

```
        For A:=0 to TotalLine-1 do
          Begin
```

Lakukan proyeksi koordinat titik pertama dari garis. Koordinat ini diperoleh dari Vertex Bank dengan index yang ditunjukkan variabel index A pada Line Bank yang sedang diakses.

```
Psv_Projection(VertexBank[LineBank[A].A].X,
       VertexBank[LineBank[A].A].Y,
             VertexBank[LineBank[A].A].Z,X1,Y1);
```

Lakukan proyeksi koordinat titik kedua dari garis. Sekali lagi koordinat ini diperoleh dari Vertex Bank dengan index yang ditunjukkan variabel index B pada Line Bank yang sedang diakses.

```
Psv_Projection(VertexBank[LineBank[A].B].X,
       VertexBank[LineBank[A].B].Y,
             VertexBank[LineBank[A].B].Z,X2,Y2);
```

Tampilkan garis pada layar monitor dengan warna yang telah ditentukan.

```
            Line(Buff^,X1,Y1,X2,Y2,Color);
          End;
      End;
  End;

Procedure Display_Object(No:Integer);
  Begin
    Wire_Frame_Projection(No);
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
```

Tampilkan ketiga objek secara terpisah dengan memanggil procedure Display_Object dengan parameter nomor objek yang hendak ditampilkan.

```
      Display_Object(0);
      Display_Object(1);
      Display_Object(2);
      Line(Buff^,160,0,160,199,15);
      Line(Buff^,0,100,319,100,15);
      MoveTo(Buff^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
  Until Ch=#27;
 End;

Begin
 Initialize;
 Main;
 DeInitialize;
End.
```

3.2 Transformasi

Dunia game programming tidak akan pernah lepas dari ilmu matematika, apalagi kalau kita bermain-main dengan tiga dimensi. Untuk itu para pembaca diharapkan benar-benar menguasai matrix. Pada bagian ini kita akan banyak membahas tentang matrix, operasi-operasinya serta hubungannya dengan dunia tiga dimensi. Para pembaca tidak perlu khawatir, penerapan ilmu matematika yang sebenarnya itu sangat

mengasyikkan. Memang pada seri ini pembahasan game programming bersifat lebih matematis dan serius, tetapi saya akan berusaha sejauh mungkin untuk membawa topik ini secara santai dan menarik.

Ok, pelajaran matematika dimulai ...

3.3 Matrix

3.3.1 Definisi Matrix

Secara umum, matrix adalah sebuah himpunan angka-angka yang disusun secara kolom dan baris. Bagi para pemrogram ... hmmm, matrix adalah struktur data array dua dimensi yang berisi angka.

Misalnya:

$$A = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 4 & 2 \\ 2 & 8 \\ 3 & 7 \end{bmatrix}$$

Matrix dapat digunakan untuk menyimpan posisi/koordinat dua dimensi maupun tiga dimensi dari suatu objek. Untuk menyimpan koordinat dua dimensi diperlukan matrix berdimensi/berukuran 2 x 1dengan format:

$$\begin{bmatrix} X \\ Y \end{bmatrix}$$

Misalnya:

Posisi suatu objek ditentukan sebagai X = 20 dan Y =40

Maka matrix posisi dari objek ialah:

$$\begin{bmatrix} 20 \\ 40 \end{bmatrix}$$

Untuk menyimpan koordinat tiga dimensi diperlukan matrix berdimensi/ berukuran 4 x 1 dengan format:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Angka 1 adalah konstanta.

3.3.2 Operasi Matrix

Penjumlahan dan pengurangan

Antara matrix yang satu dengan yang lainnya dapat dilakukan proses penambahan. Dengan syarat dimensi/ukuran dari kedua matrix yang hendak ditambahkan tersebut sama persis.

Misalnya:

$$\text{Matrix } A = \begin{bmatrix} 23 \\ 12 \\ 44 \end{bmatrix} \qquad \text{Matrix } B = \begin{bmatrix} 12 \\ 11 \\ -3 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 35 \\ 23 \\ 41 \end{bmatrix}$$

$$A\text{-}B = \begin{bmatrix} 11 \\ 1 \\ 47 \end{bmatrix}$$

Perkalian skalar

Perkalian secara skalar ialah mengalikan seluruh element matrix dengan suatu konstanta.

Misalnya:

$$A = \begin{bmatrix} 2 & 4 \\ 5 & 2 \\ 3 & 1 \end{bmatrix}$$

$$A \times 4 = \begin{bmatrix} 8 & 16 \\ 20 & 8 \\ 12 & 4 \end{bmatrix}$$

Perkalian vektor

Perkalian secara vektor ialah mengalikan sebuah matrix dengan matrix yang lain. Syaratnya ialah jumlah baris matrix yang pertama harus sama dengan jumlah kolom matrix yang kedua, juga jumlah kolom matrix yang pertama harus sama dengan jumlah baris matrix yang kedua.

Format perkalian yang harus dilakukan ialah sebagai berikut:

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \qquad B = \begin{bmatrix} g & h & I \\ j & k & l \end{bmatrix}$$

$$\text{Maka} \quad A \times B = \begin{bmatrix} ag + bj & ah + bk & aI + bl \\ cg + dj & ch + dk & cI + dl \\ eg + fj & eh + fk & eI + fl \end{bmatrix}$$

Misalnya:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 5 \\ 3 & 0 \end{bmatrix}$$

$$\text{Maka} \quad A \times B = \begin{bmatrix} 2+6 & 5+0 \\ 4+12 & 10+0 \end{bmatrix}$$

$$= \begin{bmatrix} 8 & 5 \\ 16 & 10 \end{bmatrix}$$

Hasil perkalian secara vektor antara dua buah matrix akan menghasilkan sebuah matrix dengan ukuran baris sebesar baris matrix kedua dan kolom sebesar kolom matrix pertama.

Satu lagi yang perlu diingat, perkalian matrix tidak bersifat komutatif.
Jadi A x B tidak sama dengan B x A.

3.3.3 Matrix Identitas

Ialah matrix yang jika dikalikan secara vektor terhadap suatu matrix A akan menghasilkan matrix A juga.

$$A \times I = A$$

Matrix Identitas dilambangkan sebagai I

$$\text{di mana } I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

untuk matrix 3 x 3.

Nah cukup sampai di sini pembahasan kita tentang matrix. Sekarang apa hubungan antara matrix dengan pemrograman tiga dimensi?

3.3.4 Transformasi

Transformasi ialah perubahan posisi suatu objek, baik perubahan posisi itu terjadi karena pergeseran (dilatasi) ataupun perputaran (rotasi) ataupun juga penskalaan (scaling).

Sebagaimana yang telah kita pelajari bahwa koordinat tiga dimensi dapat disimpan dalam bentuk matrix 1 x 4 dengan format:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Sedangkan operasi perputaran, pergeseran ataupun penskalaan dapat dilambangkan dengan matrix 4 x 4.

Matrix dilatasi:

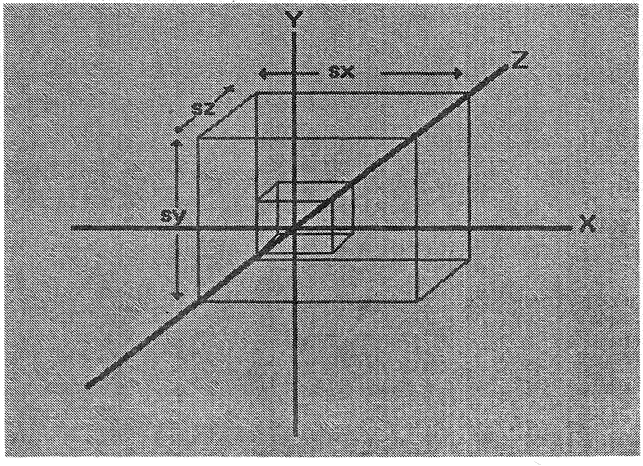
$$\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

tx adalah pergeseran pada sumbu X.
ty adalah pergeseran pada sumbu Y.
tz adalah pergeseran pada sumbu Z.

Matrix scaling:

$$\begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

sx adalah penskalaan sumbu X.
sy adalah penskalaan sumbu Y.
sz adalah penskalaan sumbu Z.



Matrix rotasi:

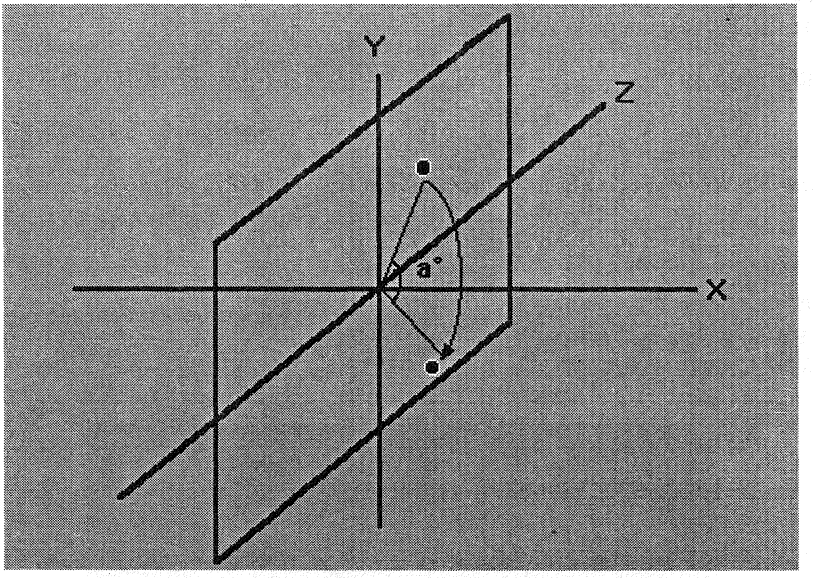
Sebagaimana yang telah kita ketahui, rotasi terhadap koordinat dua dimensi memerlukan operasi matrix 2 x 2 yaitu:

$$\begin{bmatrix} \cos a & -\sin a \\ \sin a & \cos a \end{bmatrix}$$

Pada koordinat tiga dimensi kita tidak dapat mendefinisikan rotasi tersebut ke dalam sebuah matrix karena di sini kita berhadapan dengan tiga buah koordinat plane yaitu XY, YZ dan XZ. Untuk itu kita memiliki tiga buah matrix rotasi yang terpisah.

Rotasi sumbu X

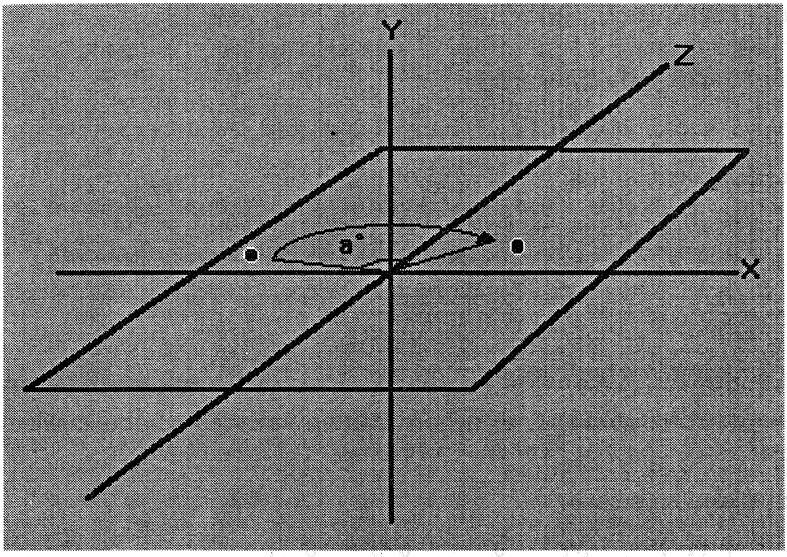
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & \sin a & 0 \\ 0 & -\sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



a adalah sudut perputaran terhadap sumbu X.

Rotasi sumbu Y

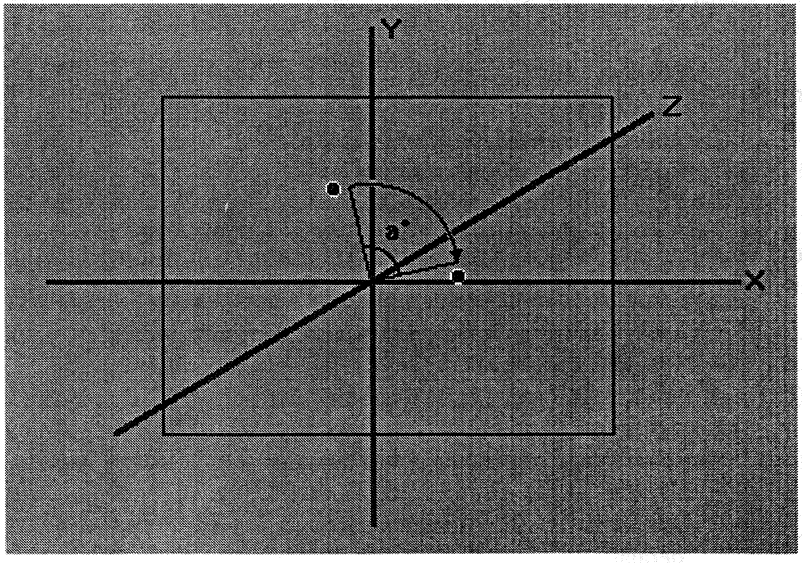
$$\begin{bmatrix} \cos a & 0 & -\sin a & 0 \\ 0 & 1 & 0 & 0 \\ \sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



a adalah sudut perputaran terhadap sumbu Y.

Rotasi sumbu Z

$$\begin{bmatrix} \text{Cos } a & \text{Sin } a & 0 & 0 \\ -\text{Sin } a & \text{Cos } a & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



a adalah sudut perputaran terhadap sumbu Z.

Untuk melakukan transformasi dari sebuah koordinat, kita tinggal mengalikan matrix yang mewakili koordinat tersebut dengan salah satu operasi matrix yang diinginkan.

Misalnya, koordinat (10 , 12 , 3) hendak di-dilatasi sejauh tx=1, ty=3, tz=-4

Maka dilakukan proses:

$$\begin{bmatrix} 10 \\ 12 \\ 3 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 11 \\ 15 \\ -1 \\ 1 \end{bmatrix}$$

Sebagai koordinat hasil transformasi ialah (11, 15, -1)

Misalnya koordinat (11, 4, 2) hendak dirotasi terhadap sumbu X sejauh 45 derajat.

Maka dilakukan proses:

$$\begin{bmatrix} 11 \\ 4 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos } 45 & \text{Sin } 45 & 0 \\ 0 & -\text{Sin } 45 & \text{Cos } 45 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hasilnya yang berupa matrix 1 x 4 akan memiliki format:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix}$$

Di mana:

X' = koordinat X hasil transformasi.
Y' = koordinat Y hasil transformasi.
Z' = koordinat Z hasil transformasi.

Untuk selanjutnya kita akan menggunakan istilah-istilah di bawah ini untuk menyingkat pembahasan kita.

P(x, y, z)	sebagai lambang matrix sebuah titik dengan koordinat (x, y, z).
D(tx, ty, tz)	sebagai lambang matrix pergeseran sejauh tx, ty, tz.
Rx(dx)	sebagai lambang matrix rotasi terhadap sumbu X sebesar dx derajat.
Ry(dy)	sebagai lambang matrix rotasi terhadap sumbu Y sebesar dy derajat.
Rz(dz)	sebagai lambang matrix rotasi terhadap sumbu Z sebesar dz derajat.
S(sx, sy, sz)	sebagai lambang matrix scalling sebesar sx, sy, sz.

3.3.5 Inverse Matrix

Inverse matrix ialah sebuah matrix yang jika dikalikan dengan matrix-nya sendiri menjadi matrix identitas.

$$\text{Inv } A \times A = I$$

Inverse matrix didapat dari 1/Determinant A x (Transpose A)

Pada matrix 2 x 2 di bawah ini:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

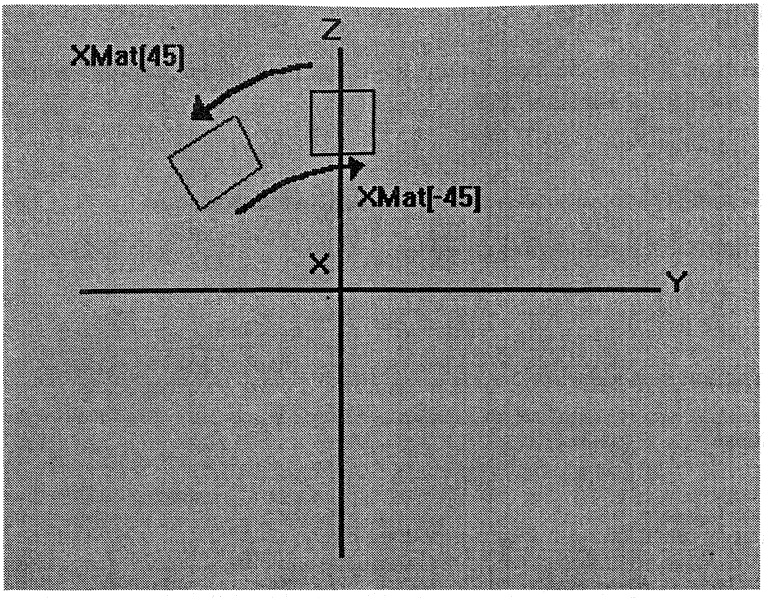
Determinant = a x d - b x c

$$\text{Transpose} = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$\text{Maka inverse} = 1/(a \times d - b \times c) \times \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Karena kita akan bermain-main di tiga dimensi maka matrix yang akan kita gunakan adalah matrix 4 x 4. Untuk itu diperlukan kalkulasi yang cukup rumit dalam mencari inverse matrixnya. Oleh karena itu kita akan mencari cara untuk mendapatkan inverse dari sebuah matrix tanpa perlu kalkulasi yang rumit. Mengingat perkalian dari matrix yang kita gunakan melambangkan suatu proses transformasi, maka inverse dari sebuah matrix dapat dilambangkan menjadi inverse dari sebuah transformasi. Hmmm, kalau begitu dapat kita cari secara geometri.

Pada matrix transformasi rotasi sumbu X dengan sudut sebesar Dx, inversenya tidak lain ialah rotasi sumbu X dengan sudut sebesar -Dx. Sederhana kan? Demikian juga dengan rotasi sumbu Z dan Y. Kita cukup mengambil sudut negatif sebagai inversenya. Karena secara geometri, jika sebuah koordinat dikalikan dengan matrix transformasi. Kemudian hasil akhirnya kita kalikan dengan matrix inverse dari transformasi, maka hasil akhirnya akan kembali menjadi koordinat awal sebelum transformasi.



Sebuah persegi panjang yang ditransformasikan dengan matrix Rotasi sumbu X 45 derajat. Inversenya ialah matrix Rotasi sumbu X -45 derajat. Karena dengan matrix ini, objek kembali ke posisi semula.

3.3.6 Matrix Concatenation

Dalam melakukan transformasi yang terdiri dari beberapa proses seperti dilatasi sejauh tx=10, ty=3, tz=1 yang dilanjutkan oleh rotasi 30 derajat terhadap sumbu X kemudian rotasi 30 derajat terhadap sumbu Z, biasanya kita mengalikan koordinat titik dengan D(10,3,1) kemudian hasil akhirnya kita kalikan lagi dengan Rx(30) dan sekali lagi hasil akhirnya kita kalikan dengan Rz(30).

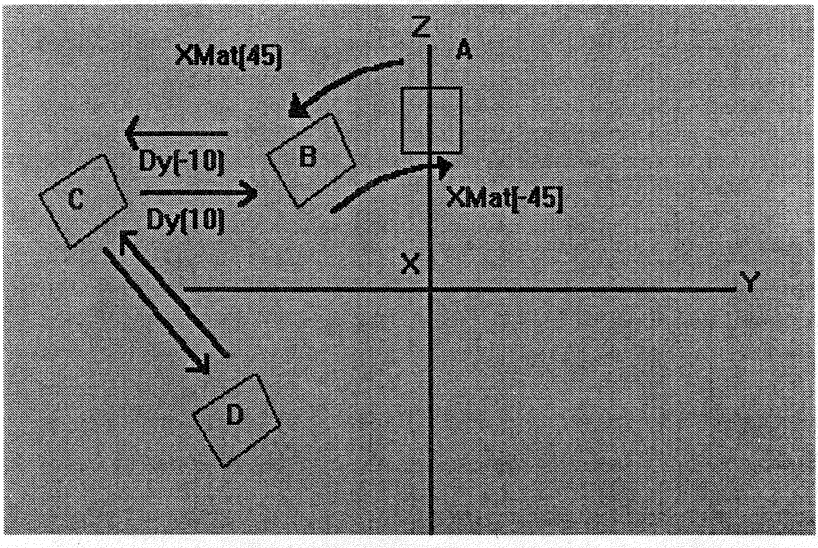
Dalam dunia matrix dikenal matrix concatenation yang berarti kita dapat menggabungkan semua matrix operasi menjadi sebuah matrix gabungan yang kemudian baru kita kalikan pada koordinat titik untuk melakukan transformasi secara serentak.

Cara menggabungkan matrix-matrix tersebut ialah dengan cara mengalikannya secara berurutan berdasarkan urutan operasi yang hendak kita lakukan. Pada contoh di atas, kita dapat mendeklarasikan sebuah matrix identitas yang kemudian dilakukan D(10,3,1) kemudian dikalikan dengan Rx(30) dan kemudian dikalikan dengan Rz(30).

Kemudian hasil akhir dari matrix concatenation ini katakanlah matrix M, dikalikan dengan koordinat titik.

Nah sekarang bagaimana mencari inverse dari matrix M tersebut? Matrix ini didapat dari hasil perkalian beberapa matrix transformasi yang dilakukan secara berturutan. Oleh karena itu inverse dari matrix ini juga terdiri dari beberapa matrix transformasi yang dikalikan, dengan syarat jika sebuah koordinat dikalikan dengan M menjadi koordinat yang baru, koordinat baru ini jika dikalikan dengan inverse matrix tadi harus kembali menjadi koordinat semula. Untuk ini kita tinggal membalik seluruh perkalian dari matrix transformasi yang dilakukan untuk mencari inverse!!! Sederhana kan?

Perhatikan diagram di bawah ini.



Inverse dari operasi rotasi sumbu X dengan sudut 45 derajat yang dilanjutkan dengan dilatasi Y sejauh -10 dan diteruskan dengan sebuah dilatasi lagi, tidak lain ialah kebalikan dari dilatasi terakhir yang dilanjutkan dengan dilatasi Y sejauh 10 dan diteruskan dengan rotasi sumbu X sebesar -45 derajat.

Dengan kata lain A >> B >> C memiliki matrix inverse -C>>-B>>-A di mana A, B dan C melambangkan sebuah operasi transformasi.

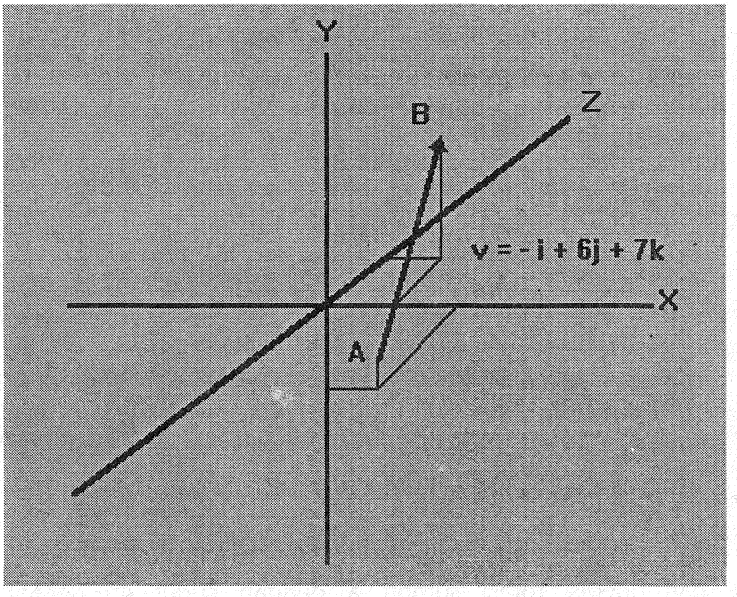
Untuk sementara kita akan tinggalkan dulu masalah matrix concatenation ini. Pada Bab 8 3D Inverse Transformation, kita akan membahasnya habis-habisan.

Keuntungan dari matrix concatenation technique ini ialah menyederhanakan proses transformasi menjadi sebuah matrix yang berisi transformasi gabungan yang akan dilakukan pada suatu titik.

Sampai di sinilah pembahasan kita mengenai matrix.

3.4 Vektor

Vektor adalah sebuah format untuk menyatakan arah dan besarnya. Vektor dalam dunia dua dimensi memiliki dua buah komponen yaitu i dan j. i adalah komponen arah horizontal (sumbu X) sedangkan j adalah komponen arah vertikal (sumbu Y). Dalam dunia tiga dimensi, vektor memiliki tiga komponen yaitu i, j dan k. Di mana tambahan komponen k adalah komponen arah kedalaman (sumbu Z).



Garis AB adalah vektor dengan komponen -i + 6j + 7k. Vektor ini dapat diartikan sebagai perpindahan dari titik awal (titik A) ke titik akhir (titik B).

Karena dari titik A ke titik B diperlukan perpindahan koordinat X sejauh -1 maka komponen i berharga -1. Karena dari titik A ke titik B diperlukan perpindahan koordinat Y sejauh 6 maka komponen j berharga 6. Demikian pula dengan komponen k yang berharga 7.

3.4.1 Magnitude Vektor

Magnitude vektor ialah besar/panjang dari vektor. Dapat juga diartikan sebagai panjang garis yang mewakili vektor tersebut. Pada diagram di atas, magnitude vektor AB ialah panjang garis AB. Berdasarkan dalil Phytagoras, Magnitude AB ialah:

$$|AB| = SQRT(a^2 + b^2 + c^2)$$

Di mana a ialah harga komponen i, b ialah harga komponen j dan c adalah harga komponen k.

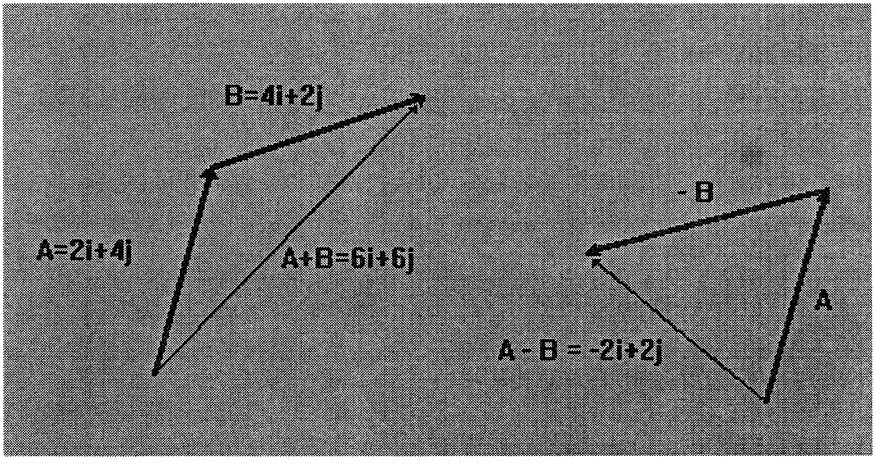
$$|AB| = SQRT(1 + 36 + 49)$$
$$= SQRT(86)$$

3.4.2 Penambahan dan Pengurangan Vektor

Penambahan dan pengurangan antara dua vektor dilakukan dengan melakukan operasi penambahan atau pengurangan pada masing-masing komponennya. Misalnya Vektor A = 2i + 2j akan ditambahkan dengan Vektor B = 4i + 2j.

A + B = (2+4)i + (2+2)j = 6i + 4j.

Di sini kita menggunakan vektor dua dimensi, sebagai contoh agar kita lebih mudah memvisualisasikannya.



Secara geometris, penambahan dan pengurangan vektor dapat digambarkan sebagai transformasi dilatasi yang dilanjutkan dengan transformasi dilatasi yang lain. Pada A-B kita tinggal berpindah dengan vektor A kemudian lanjutkan perjalanan ke arah yang berlawanan dengan vektor B karena dilakukan pengurangan dengan B.

3.4.3 Unit Vektor

Unit Vektor dari suatu vektor ialah Vektor lain yang memiliki magnitude = 1 tetapi arahnya masih tetap sama. Setiap vektor memiliki unit vektor. Untuk mencari unit vektor dari suatu vektor ialah dengan membagi masing-masing

komponennya dengan magnitude-nya.

Misalnya: A = 2i + 4j + k

Unit Vektor A = 2/SQRT(21)i + 4/SQRT(21)j + 1/SQRT(21)k

Untuk membuktikan bahwa unit vektor ini memiliki magnitude =1 cobalah hitung magnitude dari unit vektor A.

Magnitude Unit Vektor A = SQRT(a^2 + b^2 + c^2)

Di mana a = 2/SQRT(21), b = 4/SQRT(21) dan c = 1/SQRT(21)

Magnitude Unit Vektor A = SQRT(4/21 + 16/21 + 1/21)

= SQRT (1)

= 1

Terbukti.

3.4.4 Dot Product

Dot product dilakukan dengan cara mengalikan masing-masing komponen dari dua buah vektor.

Lambang operasi dari dot product ini ialah titik.

Misalnya: Vektor A = 2i + 3j dan Vektor B = i + 4j.

A . B = (2*1) + (3*4)

= 2 + 12

= 14

Dot product ini biasanya dilakukan untuk menghitung sudut antara dua buah vektor yang memiliki rumus:

a = arCos [(A . B) / (| A | * | B |)

Di mana a ialah sudut antara dua Vektor.

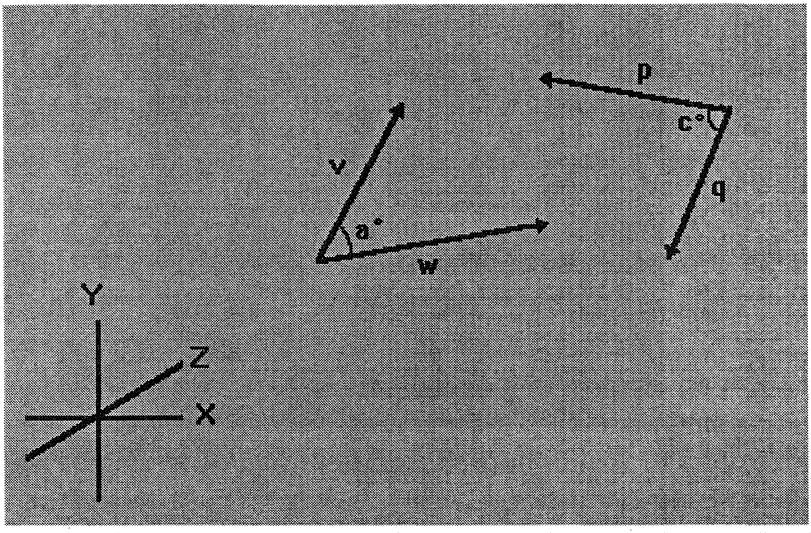
Yang perlu diingat ialah, jika A . B:

> 0 maka sudut antara dua vektor ialah sudut acute (lebih kecil dari 90 derajat)

< 0 maka sudut antara dua vektor ialah sudut obtuse (lebih besar dari 90 derajat)

= 0 maka sudut antara dua vektor ialah sudut 90 derajat.

Kesimpulan ini akan kita pergunakan untuk melakukan proses BACKFACE FILTERING yang akan kita bahas pada bab-bab selanjutnya.



Pada diagram di atas, sudut antara v dan w ditemukan = a derajat. Sedangkan sudut antara p dan q ditemukan = c derajat.

3.4.5 Cross Product

Cross product ialah mengalikan dua buah vektor untuk memperoleh sebuah vektor baru yang tegak lurus dengan kedua vektor tersebut. Vektor ini biasa disebut dengan normal vektor.

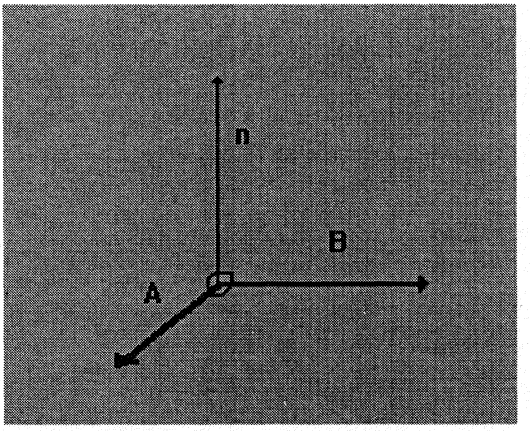
Cross Product dilambangkan dengan x.

Rumus Cross Product ialah sebagai berikut:

A x B = (bA * cB-cA * bB)I - (aA * cB-cA * aB)j + (aA * bB-bA * aB)k

Di mana:

aA	=	Komponen i dari vektor A
bA	=	Komponen j dari vektor A
cA	=	Komponen k dari vektor A
aB	=	Komponen i dari vektor B
bB	=	Komponen j dari vektor B
cB	=	Komponen k dari vektor B



Cross Product ini akan memegang peranan penting dalam mencari normal vektor dari sebuah poligon.

Selesailah pembahasan kita tentang vektor.

3.5 Perakitan Unit MATH.TPU

Sekarang kita akan mencoba untuk merakit sebuah unit program untuk menangani matrix dan vektor.

```
{-------------------------------------------------------------}
{ Program     : MATH.PAS                                      }
{ Coder       : Agustinus Nalwan                              }
{ Description: Unit untuk melakukan perhitungan vektor dan matrix }
{-------------------------------------------------------------}
Unit Math;

Interface

Uses Crt,Dos;

Type Mat4x4 = array[0..3,0..3] of Real;
     Mat1x4 = array[0..3] of Real;
     Vec3D  = array[0..2] of Real;

Function DToR(Deg:Real) : Real;
Function RToD(Rad:Real) : Real;
Function Cosinus(Deg:Real) : Real;
Function Sinus(Deg:Real)   : Real;
Function Tangent(Deg:Real) : Real;
Function arCos(Num:Real) : Real;
Function arSin(Num:Real) : Real;
Function arTan(Num:Real) : Real;

Procedure Mat_Identity(Var A:Mat4x4);
Procedure Mat_XRotate(Var A:Mat4x4;Deg:Real);
Procedure Mat_YRotate(Var A:Mat4x4;Deg:Real);
```

```
Procedure Mat_ZRotate(Var A:Mat4x4;Deg:Real);
Procedure Mat_Dilatasi(Var A:Mat4x4;tx,ty,tz:Real);
Procedure Mat_Scaling(Var A:Mat4x4;sx,sy,sz:Real);
Procedure Mat_Multiply4x4(Var A:Mat4x4;B:Mat4x4);
Procedure Mat_Multiply4x1(A:Mat4x4;Var B:Mat1x4);
Procedure Mat_Add(Var A:Mat4x4;B:Mat4x4);
Procedure Mat_Subtract(Var A:Mat4x4;B:Mat4x4);

Procedure Vect_Add(Var A:Vec3D;B:Vec3D);
Procedure Vect_Sub(Var A:Vec3D;B:Vec3D);
Procedure Vect_Unit(Var A:Vec3D;B:Vec3D);
Function  Vect_Mag(A:Vec3D) : Real;
Function  Vect_Dot(A,B:Vec3D) : Real;
Function  Vect_Angle(A,B:Vec3D) : Real;
Procedure Vect_Cross(Var A:Vec3D;B,C:Vec3D);

Implementation

Function DToR(Deg:Real) : Real;
  Begin
    DToR:=(Deg/180)*pi;
  End;

Function RToD(Rad:Real) : Real;
  Begin
    RToD:=(Rad/pi)*180;
  End;

Function Cosinus(Deg:Real) : Real;
  Begin
    Cosinus:=Cos(DToR(Deg));
  End;

Function Sinus(Deg:Real)    : Real;
  Begin
    Sinus:=Sin(DToR(Deg));
  End;

Function Tangent(Deg:Real)    : Real;
  Begin
    Tangent:=Sinus(Deg)/Cosinus(Deg);
  End;

Function arCos(Num:Real) : Real;
  Begin
    If Num=0 then arCos:=pi/2 else
      arCos:=ArcTan (sqrt (1-sqr (Num)) /Num);
  End;

Function arSin(Num:Real) : Real;
  Begin
    If Num=1 then arSin:=pi/2 else
      arSin:=ArcTan (Num/sqrt (1-sqr (Num)));
  End;

Function arTan(Num:Real) : Real;
  Begin
```

```
    arTan:=ArcTan(Num);
  End;

Procedure Mat_Identity(Var A:Mat4x4);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_XRotate(Var A:Mat4x4;Deg:Real);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=Cosinus(Deg);A[1,2]:=Sinus(Deg);A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=-Sinus(Deg);A[2,2]:=Cosinus(Deg);A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_YRotate(Var A:Mat4x4;Deg:Real);
  Begin
    A[0,0]:=Cosinus(Deg);A[0,1]:=0;A[0,2]:=-Sinus(Deg);A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=Sinus(Deg);A[2,1]:=0;A[2,2]:=Cosinus(Deg);A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_ZRotate(Var A:Mat4x4;Deg:Real);
  Begin
    A[0,0]:=Cosinus(Deg);A[0,1]:=Sinus(Deg);A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=-Sinus(Deg);A[1,1]:=Cosinus(Deg);A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_Dilatasi(Var A:Mat4x4;tx,ty,tz:Real);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=tx;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=ty;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=tz;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_Scaling(Var A:Mat4x4;sx,sy,sz:Real);
  Begin
    A[0,0]:=sx;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=sy;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=sz;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_Multiply4x4(Var A:Mat4x4;B:Mat4x4);
Var d,e,c:Integer;
    BuffMat:Mat4x4;
  Begin
    For d:=0 to 3 do
      For e:=0 to 3 do
        Begin
```

```
          BuffMat[e,d]:=0;
          For c:=0 to 3 do
            BuffMat[e,d]:=BuffMat[e,d]+A[e,c]*B[c,d];
        End;
    A:=BuffMat;
  End;

Procedure Mat_Multiply4x1(A:Mat4x4;Var B:Mat1x4);
Var e,c:Integer;
    BuffMat:Mat1x4;
  Begin
    For e:=0 to 3 do
      Begin
        BuffMat[e]:=0;
        For c:=0 to 3 do
          BuffMat[e]:=BuffMat[e]+A[e,c]*B[c];
      End;
    B:=BuffMat;
  End;

Procedure Mat_Add(Var A:Mat4x4;B:Mat4x4);
Var c,d:Integer;
  Begin
    For c:=0 to 3 do
      For d:=0 to 3 do
        A[c,d]:=A[c,d]+B[c,d];
  End;

Procedure Mat_Subtract(Var A:Mat4x4;B:Mat4x4);
Var c,d:Integer;
  Begin
    For c:=0 to 3 do
      For d:=0 to 3 do
        A[c,d]:=A[c,d]-B[c,d];
  End;

Procedure Vect_Add(Var A:Vec3D;B:Vec3D);
Var c:Integer;
  Begin
    For c:=0 to 2 do
      A[c]:=A[c]+B[c];
  End;

Procedure Vect_Sub(Var A:Vec3D;B:Vec3D);
Var c:Integer;
  Begin
    For c:=0 to 2 do
      A[c]:=A[c]-B[c];
  End;

Function Vect_Mag(A:Vec3D) : Real;
  Begin
    Vect_Mag:=Sqrt(A[0]*A[0]+A[1]*A[1]+A[2]*A[2]);
  End;

Procedure Vect_Unit(Var A:Vec3D;B:Vec3D);
Var M:Real;
  Begin
```

```
    M:=Vect_Mag(B);
    A[0]:=B[0]/M;
    A[1]:=B[1]/M;
    A[2]:=B[2]/M;
  End;

Function  Vect_Dot(A,B:Vec3D) : Real;
  Begin
    Vect_Dot:=A[0]*B[0]+A[1]*B[1]+A[2]*B[2];
  End;

Function  Vect_Angle(A,B:Vec3D) : Real;
  Begin
   Vect_Angle:=RToD(arCos(Vect_Dot(A,B)/(Vect_Mag(A)*Vect_Mag(B))));
  End;

Procedure Vect_Cross(Var A:Vec3D;B,C:Vec3D);
  Begin
    A[0]:=B[1]*C[2]-B[2]*C[1];
    A[1]:=-B[0]*C[2]+B[2]*C[0];
    A[2]:=B[0]*C[1]-B[1]*C[0];
  End;

Begin
End.
```

PENJELASAN LISTING PROGRAM

Program di atas adalah program utama yang mendasari tiga dimensi transformasi. Hampir seluruh fungsi yang akan kita pergunakan pada pemrograman tiga dimensi akan berkenaan dengan unit ini. Oleh karena itu pemahaman cara kerja unit ini sangatlah penting.

Pertama-tama marilah kita lihat data structure yang dipergunakan.

Mat4x4 ialah tipe struktur data yang akan kita pergunakan untuk menyimpan matrix transformasi tiga dimensi yang berukuran 4 x 4. Untuk itu kita gunakan array dua dimensi berukuran 4 x 4. Adapun tipe dari masing-masing elemennya ialah Real.

```
Type
     Mat4x4 = array[0..3,0..3] of Real;
```

Untuk menyimpan sebuah koordinat tiga dimensi, diperlukan matrix 1 x 4 seperti yang telah kita pelajari di atas. Untuk itu dideklarasikan array satu dimensi berukuran 4 dan bertipe Real untuk masing-masing elemennya.

```
     Mat1x4 = array[0..3] of Real;
```

Untuk vektor, kita gunakan array satu dimensi berukuran 3. Cell pertama akan digunakan untuk menyimpan komponen i, cell kedua untuk komponen j dan cell ketiga untuk komponen k dari vektor.

```
     Vec3D  = array[0..2] of Real;
```

Sekarang kita akan membahas fungsi-fungsi yang disediakan oleh unit ini satu persatu.

NAMA : **Function DToR**
FUNGSI : Untuk meng-convert sudut dari degree ke radiant.
Hal ini dilakukan karena komputer hanya mengenal sistem radiant.
PARAMETER: Deg = Real
Sudut dalam degree yang akan di-convert.
OUTPUT : Hasil convert dari sudut dalam radiant.

PENJELASAN:

```
Function DToR(Deg:Real) : Real;
  Begin
```

Secara matematis, karena pi atau 22/7 radiant sama dengan 180 derajat maka untuk memperoleh sudut dalam radiant, suatu sudut dalam degree harus dikalikan dengan pi/180.

NAMA : **Function RToD**
FUNGSI : Untuk meng-convert sudut dari radiant ke degree.
Hal ini dilakukan karena komputer hanya mengenal sistem radiant.
PARAMETER: Rad = Real
Sudut dalam radiant yang akan di-convert.
OUTPUT : Hasil convert dari sudut dalam degree.

PENJELASAN:

```
Function RToD(Rad:Real) : Real;
  Begin
```

Sebaliknya, untuk mengembalikan dari Radiant ke degree dilakukan perkalian dengan 180/pi.

```
    RToD:=(Rad/pi)*180;
  End;
```

NAMA : **Function Cosinus**
FUNGSI : Untuk menghasilkan Cosinus dari suatu sudut dalam degree.
PARAMETER: Deg = Real
Sudut dalam degree yang akan dicari Cosinus-nya.
OUTPUT : Cosinus dari sudut tersebut.

PENJELASAN:

```
Function Cosinus(Deg:Real) : Real;
  Begin
```

Dilakukan konversi ke radiant dengan memanggil fungsi DToR sebelum parameter Deg dikirimkan ke fungsi Cos. Dengan ini maka parameter yang dikirimkan lewat Cos akan bertipe Radiant.

```
    Cosinus:=Cos(DToR(Deg));
  End;
```

NAMA : **Function Sinus**
FUNGSI : Untuk menghasilkan Sinus dari suatu sudut dalam degree.
PARAMETER: Deg = Real
Sudut dalam degree yang akan dicari Sinus-nya.
OUTPUT : Sinus dari sudut tersebut.

PENJELASAN:

```
Function Sinus(Deg:Real)    : Real;
  Begin
```

Dilakukan konversi ke radiant dengan memanggil fungsi DToR sebelum parameter Deg dikirimkan ke fungsi Sin. Dengan ini maka parameter yang dikirimkan lewat Sin akan bertipe Radiant.

```
    Sinus:=Sin(DToR(Deg));
  End;
```

NAMA : **Function Tangent**
FUNGSI : Untuk menghasilkan Tangent dari suatu sudut dalam degree.
PARAMETER: Deg = Real
Sudut dalam degree yang akan dicari Tangent-nya.
OUTPUT : Tangent dari sudut tersebut.

PENJELASAN:

```
Function Tangent(Deg:Real)    : Real;
  Begin
```

Karena tidak adanya fungsi Tangent pada compiller Turbo Pascal, maka secara matematis, Tangent dihasilkan dengan membagi Sinus dengan Cosinus.

```
    Tangent:=Sinus(Deg)/Cosinus(Deg);
  End;
```

NAMA : **Function arCos**
FUNGSI : Untuk menghasilkan Inverse Cosinus dari suatu harga.
PARAMETER: Num = Real
Harga yang hendak dicari Inverse Cosinus-nya.
OUTPUT : Inverse Cosinus dari Num dalam radiant.

PENJELASAN:

```
Function arCos(Num:Real) : Real;
  Begin
```

Jika Num = 0 maka Inverse Cosinus-nya ialah 0. Jika Num tidak sama dengan 0 maka lakukan proses di bawah ini. Inverse Cosinus ini dilakukan dengan perhitungan trigonometri karena tidak adanya fungsi Inverse Cosinus pada compiller Turbo Pascal.

Pengecekan Num = 0 pada awal program dilakukan untuk menghindari pembagian dengan 0.

Pembuktian lebih lanjut (bagi para pecinta matematika tentu ingin tahu khan?), dari rumus trigonometri yang dipergunakan di bawah ini ialah sebagai berikut:

$a = ArcTan(Tan(a))$

$a = ArcTan(Sin(a)/Cos(a))$

$Sin(a)=sqrt(1-Cos(a)\ ^\ 2)$

$a = ArcTan(sqrt(1-Cos(a)\ ^\ 2)/Cos(a))$

$Cos(a) = Num$

$a = ArcTan(sqrt(1-Num\ ^\ 2)/Num)$

Terbukti!!!

```
    If Num=0 then arCos:=pi/2 else
      arCos:=ArcTan (sqrt (1-sqr (Num)) /Num);
  End;
```

NAMA : **Function arSin**
FUNGSI : Untuk menghasilkan Inverse Sinus dari suatu harga.
PARAMETER: Num = Real
Harga yang hendak dicari Inverse Sinus-nya.
OUTPUT : Inverse Sinus dari Num dalam radiant.

PENJELASAN:

```
Function arSin(Num:Real) : Real;
  Begin
```

Jika Num=1 maka Inverse Sinusnya ialah pi/2, jika tidak maka lakukan kembali kalkulasi trigonometri. Pembuktian kali ini cukup sederhana karena untuk mencari Inverse Sinus, kita cukup membalik Num dan sqrt(1-sqr(Num)). Bagi para pecinta matematika, cobalah teruskan pembuktian rumus Inverse Sinus tersebut.

```
    If Num=1 then arSin:=pi/2 else
      arSin:=ArcTan (Num/sqrt (1-sqr (Num)));
  End;
```

NAMA : **Function arTan**
FUNGSI : Untuk menghasilkan Inverse Tangent dari suatu harga.
PARAMETER: Num = Real
Harga yang hendak dicari Inverse Tangent-nya.
OUTPUT : Inverse Tangent dari Num dalam radiant.

PENJELASAN:

```
Function arTan(Num:Real) : Real;
  Begin
```

Untuk Inverse Tangent, sangat sederhana karena kita cukup memanggil fungsi ArcTan dari Turbo Pascal.

```
    arTan:=ArcTan(Num);
  End;
```

Sekarang kita mulai pembahasan dari fasilitas-fasilitas Matrix yang disediakan.

NAMA : **Procedure Mat_Identity**
FUNGSI : Untuk mengisi sebuah matrix menjadi matrix identitas.
PARAMETER: A = Mat4x4
Matrix yang hendak dijadikan matrix identitas.
OUTPUT : Matrix A menjadi Matrix Identitas.

PENJELASAN:

```
Procedure Mat_Identity(Var A:Mat4x4);
  Begin
```

Berdasarkan ilmu matrix yang telah kita pelajari, Array dua dimensi yang mewakili matrix 4 x 4 kita isi elemennya dengan aturan matrix identitas.

```
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;
```

NAMA : **Procedure Mat_XRotate**
FUNGSI : Untuk mengisi sebuah matrix menjadi matrix Rotasi terhadap sumbu X dengan sudut rotasi sebesar Deg dalam degree.
PARAMETER: A = Mat4x4
Matrix yang hendak dijadikan matrix Rotasi sumbu X.

Deg = Real
Sudut rotasi dari Matrix dalam degree.

OUTPUT : Matrix A menjadi matrix Rotasi terhadap sumbu X dengan sudut rotasi Deg.

PENJELASAN:

Isikan matrix dengan menuruti aturan matrix Rotasi terhadap sumbu X. Lakukan pemanggilan fungsi Sinus dan Cosinus yang telah dirancang.

```
Procedure Mat_XRotate(Var A:Mat4x4;Deg:Real);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=Cosinus(Deg);A[1,2]:=Sinus(Deg);A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=-Sinus(Deg);A[2,2]:=Cosinus(Deg);A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;
```

NAMA : **Procedure Mat_YRotate**

FUNGSI : Untuk mengisi sebuah matrix menjadi matrix Rotasi terhadap sumbu Y dengan sudut rotasi sebesar Deg dalam degree.

PARAMETER: A = Mat4x4
Matrix yang hendak dijadikan matrix Rotasi sumbu Y.
Deg = Real
Sudut rotasi dari Matrix dalam degree.

OUTPUT : Matrix A menjadi matrix Rotasi terhadap sumbu Y dengan sudut rotasi Deg.

PENJELASAN:

Isikan matrix dengan menuruti aturan matrix Rotasi terhadap sumbu Y. Lakukan pemanggilan fungsi Sinus dan Cosinus yang telah dirancang.

```
Procedure Mat_YRotate(Var A:Mat4x4;Deg:Real);
  Begin
    A[0,0]:=Cosinus(Deg);A[0,1]:=0;A[0,2]:=-Sinus(Deg);A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=Sinus(Deg);A[2,1]:=0;A[2,2]:=Cosinus(Deg);A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;
```

NAMA : **Procedure Mat_ZRotate**

FUNGSI : Untuk mengisi sebuah matrix menjadi matrix Rotasi terhadap sumbu Z dengan sudut rotasi sebesar Deg dalam degree.

PARAMETER: A = Mat4x4
Matrix yang hendak dijadikan matrix Rotasi sumbu Z.

Deg = Real
Sudut rotasi dari Matrix dalam degree.

OUTPUT : Matrix A menjadi matrix Rotasi terhadap sumbu Z dengan sudut rotasi Deg.

PENJELASAN:

Isikan matrix dengan menuruti aturan matrix Rotasi terhadap sumbu Z. Lakukan pemanggilan fungsi Sinus dan Cosinus yang telah dirancang.

```
Procedure Mat_ZRotate(Var A:Mat4x4;Deg:Real);
  Begin
    A[0,0]:=Cosinus(Deg);A[0,1]:=Sinus(Deg);A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=-Sinus(Deg);A[1,1]:=Cosinus(Deg);A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;
```

NAMA : Procedure Mat_Dilatasi

FUNGSI : Untuk mengisi sebuah matrix menjadi matrix dilatasi terhadap sumbu X, Y dan Z sejauh tx, ty dan tz.

PARAMETER: A = Mat4x4
Matrix yang hendak dijadikan matrix Dilatasi.
Tx = Real
Jumlah dilatasi sumbu X.
ty = Real
Jumlah dilatasi sumbu Y.
tz = Real
Jumlah dilatasi sumbu Z.

OUTPUT : Matrix A menjadi matrix Dilatasi.

PENJELASAN:

Isikan matrix dengan menuruti aturan matrix Dilatasi pada sumbu X, Y dan Z.

```
Procedure Mat_Dilatasi(Var A:Mat4x4;tx,ty,tz:Real);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=tx;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=ty;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=tz;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;
```

NAMA : Procedure Mat_Scaling

FUNGSI : Untuk mengisi sebuah matrix menjadi matrix Scaling terhadap sumbu X, Y dan Z dengan skala perbesaran sebesar

sx, sy dan sz.

PARAMETER: A = Mat4x4
Matrix yang hendak dijadikan matrix Rotasi sumbu Y.
sx = Real
Skala perbesaran sumbu X.
sy = Real
Skala perbesaran sumbu Y.
sz = Real
Skala perbesaran sumbu Z.

OUTPUT : Matrix A menjadi matrix Scaling.

PENJELASAN:

Isikan matrix dengan menuruti aturan matrix Scaling terhadap sumbu X, Y dan Z.

```
Procedure Mat_Scaling(Var A:Mat4x4;sx,sy,sz:Real);
  Begin
    A[0,0]:=sx;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=sy;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=sz;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;
```

NAMA : Procedure Mat_Multiply4x4

FUNGSI : Untuk mengalikan matrix berukuran 4 x 4 dengan matrix lain yang juga berukuran 4 x 4.

PARAMETER: A = Mat4x4
Salah satu matrix yang hendak dikalikan.
B = Mat4x4
Salah satu matrix yang hendak dikalikan.

OUTPUT : Matrix A menjadi matrix hasil kali dari keduanya.

PENJELASAN:

```
Procedure Mat_Multiply4x4(Var A:Mat4x4;B:Mat4x4);
Var d,e,c:Integer;
```

Sediakan Buffer Matrix untuk menampung hasil perkalian.

```
    BuffMat:Mat4x4;
  Begin
    For d:=0 to 3 do
      For e:=0 to 3 do
        Begin
```

Set harga awal BuffMat[e,d] = 0.

```
        BuffMat[e,d]:=0;
        For c:=0 to 3 do
```

Tambahkan BuffMat[e,d] dengan hasil kali dari matrix A dengan matrix B pada baris yang sedang diakses.

```
          BuffMat[e,d]:=BuffMat[e,d]+A[e,c]*B[c,d];
      End;
```

Pindahkan isi BuffMat ke matrix A sebagai output.

```
   A:=BuffMat;
End;
```

NAMA : **Procedure Mat_Multiply4x1**
FUNGSI : Untuk mengalikan matrix berukuran 4 x 4 dengan matrix lain yang berukuran 1 x 4.
PARAMETER: A = Mat4x4
Salah satu matrix yang hendak dikalikan.
B = Mat1x4
Salah satu matrix yang hendak dikalikan.
OUTPUT : Matrix B menjadi matrix hasil kali dari keduanya.

PENJELASAN:

```
Procedure Mat_Multiply4x1(A:Mat4x4;Var B:Mat1x4);
Var e,c:Integer;
```

Buffer yang diperlukan ialah buffer matrix 1 x 4.

```
   BuffMat:Mat1x4;
Begin
```

Karena matrix kedua hanya satu dimensi maka cukup diperlukan dua buah loop.

```
   For e:=0 to 3 do
     Begin
```

Set harga awal BuffMat[e] = 0.

```
       BuffMat[e]:=0;
       For c:=0 to 3 do
```

Terus tambahkan BuffMat dengan hasil kali antara element matrix A dan element matrix B.

```
         BuffMat[e]:=BuffMat[e]+A[e,c]*B[c];
     End;
```

Pindahkan hasil perkalian ke matrix B sebagai Output.

```
   B:=BuffMat;
End;
```

NAMA : **Procedure Mat_Add**
FUNGSI : Untuk menambahkan matrix berukuran 4 x 4 dengan matrix lain yang juga berukuran 4 x 4.
PARAMETER: A = Mat4x4
Salah satu matrix yang hendak ditambahkan.
B = Mat4x4
Salah satu matrix yang hendak ditambahkan.
OUTPUT : Matrix A menjadi matrix hasil penambahan dari keduanya.

PENJELASAN:

```
Procedure Mat_Add(Var A:Mat4x4;B:Mat4x4);
Var c,d:Integer;
  Begin
```

Lakukan double counter untuk mengakses seluruh elemen dari matrix.

```
    For c:=0 to 3 do
      For d:=0 to 3 do
```

Tambahkan elemen matrix pertama dengan elemen matrix kedua pada baris dan kolom yang sama.

```
        A[c,d]:=A[c,d]+B[c,d];
  End;
```

NAMA : **Procedure Mat_Subtract**
FUNGSI : Untuk mengurangi matrix berukuran 4 x 4 dengan matrix lain yang juga berukuran 4 x 4.
PARAMETER: A = Mat4x4
Matrix yang hendak dikurangi.
B = Mat4x4
Matrix yang berfungsi sebagai pengurang.
OUTPUT : Matrix A menjadi matrix hasil pengurangan.

PENJELASAN:

```
Procedure Mat_Subtract(Var A:Mat4x4;B:Mat4x4);
Var c,d:Integer;
  Begin
```

Lakukan double counter untuk mengakses seluruh elemen matrix.

```
    For c:=0 to 3 do
      For d:=0 to 3 do
```

Kurangi elemen matrix pertama dengan elemen matrix kedua pada baris dan kolom yang sama.

```
      A[c,d]:=A[c,d]-B[c,d];
  End;
```

Fungsi-fungsi yang akan dibahas berikutnya ialah fungsi-fungsi yang berkenaan dengan vektor.

NAMA : **Procedure Vec_Add**
FUNGSI : Untuk menambah sebuah vektor dengan vektor yang lain.
PARAMETER: A = Vec3D.
Salah satu vektor yang hendak ditambahkan.
B = Vec3D.
Salah satu vektor yang hendak ditambahkan.
OUTPUT : Vektor A sebagai hasil penambahan antara keduanya.

PENJELASAN:

```
Procedure Vect_Add(Var A:Vec3D;B:Vec3D);
Var c:Integer;
  Begin
```

Lakukan loop untuk mengakses komponen i, j dan k dari vektor.

```
    For c:=0 to 2 do
```

Tambahkan komponen yang sama dari dua buah vektor.

```
      A[c]:=A[c]+B[c];
  End;
```

NAMA : **Procedure Vec_Sub**
FUNGSI : Untuk mengurangi sebuah vektor dengan vektor yang lain.
PARAMETER: A = Vec3D.
Vektor yang hendak dikurangi.
B = Vec3D.
Vektor yang digunakan untuk mengurangi.
OUTPUT : Vektor A sebagai hasil pengurangan.

PENJELASAN:

```
Procedure Vect_Sub(Var A:Vec3D;B:Vec3D);
Var c:Integer;
  Begin
```

Lakukan loop untuk mengakses komponen i, j dan k dari kedua vektor.

```
    For c:=0 to 2 do
```

Tambahkan kedua komponen yang sama dari kedua vektor.

```
      A[c]:=A[c]-B[c];
  End;
```

NAMA : **Function Vec_Mag**
FUNGSI : Untuk mencari magnitude dari sebuah vektor.
PARAMETER: A = Vec3D.
Vektor yang hendak dicari magnitude-nya.
OUTPUT : Magnitude dari vektor A.

PENJELASAN:

```
Function Vect_Mag(A:Vec3D) : Real;
  Begin
```

Gunakan rumus perhitungan jarak Distance = Sqrt (A^2 + B^2 + C^2)
Di mana A ialah komponen i dari vektor yang disimpan dalam A[0].
Di mana B ialah komponen j dari vektor yang disimpan dalam A[1].
Di mana C ialah komponen k dari vektor yang disimpan dalam A[2].

```
    Vect_Mag:=Sqrt(A[0]*A[0]+A[1]*A[1]+A[2]*A[2]);
  End;
```

NAMA : **Function Vec_Unit**
FUNGSI : Untuk mencari unit vektor dari sebuah vektor.
PARAMETER: B = Vec3D.
Vektor yang hendak dicari unit vektornya.
A = Vec3D.
Unit Vektor yang ditemukan.
OUTPUT : A sebagai unit vektor dari vektor B.

PENJELASAN:

```
Procedure Vect_Unit(Var A:Vec3D;B:Vec3D);
Var M:Real;
  Begin
```

Lakukan pemanggilan fungsi Magnitude untuk mencari magnitude dari vektor B.

```
    M:=Vect_Mag(B);
```

Bagi masing-masing komponen vektor B dengan Magnitude.

```
    A[0]:=B[0]/M;
    A[1]:=B[1]/M;
    A[2]:=B[2]/M;
  End;
```

NAMA : **Function Vec_Dot**
FUNGSI : Untuk mencari Dot Product dari dua buah vektor.
PARAMETER: A = Vec3D.
Vektor pertama untuk dicari dot product-nya.

B = Vec3D.
Vektor kedua untuk dicari dot product-nya.

OUTPUT : Hasil Dot Product antara vektor A dan B.

PENJELASAN:

```
Function  Vect_Dot(A,B:Vec3D) : Real;
  Begin
```

Dot Product diperoleh dengan menambahkan hasil perkalian antara komponen yang sama antara kedua vektor.

```
    Vect_Dot:=A[0]*B[0]+A[1]*B[1]+A[2]*B[2];
  End;
```

NAMA : Function Vect_Angle

FUNGSI : Untuk mencari sudut antara dua vektor.

PARAMETER: A = Vec3D.
Vektor pertama untuk pencarian sudut.
B = Vec3D.
Vektor kedua untuk pencarian sudut.

OUTPUT : Sudut antara vektor A dan B dalam degree.

PENJELASAN:

```
Function  Vect_Angle(A,B:Vec3D) : Real;
  Begin
```

Gunakan rumus pencarian sudut antara dua vektor seperti yang telah dijelaskan di atas. Kemudian panggil fungsi RToD untuk mengkonversi sudut dari radiant ke degree. Karena kita telah memiliki fungsi-fungsi seperti Vect_Mag, Vec_Dot, arCos maka perhitungan Dot Product menjadi sangat sederhana dan mudah dibaca karena formatnya hampir sama seperti rumus aslinya yang telah kita bahas.

```
    Vect_Angle:=RToD(arCos(Vect_Dot(A,B)/(Vect_Mag(A)*Vect_Mag(B))));
  End;
```

NAMA : Function Vec_Cross

FUNGSI : Untuk mencari vektor hasil Cross Product dari dua buah vektor.

PARAMETER: B = Vec3D.
Vektor pertama untuk pencarian Cross Product.
C = Vec3D.
Vektor kedua untuk pencarian Cross Product.
A = Vec3D.
Vektor hasil dari Cross Product.

OUTPUT : Vektor A sebagai hasil Cross Product dari vektor B dan C.

PENJELASAN:

```
Procedure Vect_Cross(Var A:Vec3D;B,C:Vec3D);
  Begin
```

Lakukan perkalian sesuai dengan cara pencarian cross product seperti yang telah kita bahas.

A x B = (A[1] * B[2]-A[2] * B[1])i-(A[0] * B[2]-A[2] * B[0])j + (A[0] * B[1]-A[1] * B[0])k

Di mana:

A[0] = Komponen i dari vektor A
A[1] = Komponen j dari vektor A
A[2] = Komponen k dari vektor A
B[0] = Komponen i dari vektor B
B[1] = Komponen j dari vektor B
B[2] = Komponen k dari vektor B

```
    A[0]:=B[1]*C[2]-B[2]*C[1];
    A[1]:=-B[0]*C[2]+B[2]*C[0];
    A[2]:=B[0]*C[1]-B[1]*C[0];
  End;

Begin
End.
```

Akhirnya, selesailah pembahasan matematika. Ingat, pemrogram sejati tidak takut matematika!!! Matematika sebenarnya sangat menyenangkan kalau kita bisa memanfaatkannya untuk sesuatu hal yang kita perlukan (plus kalau kita mengerti). Seperti contohnya dalam pembuatan game tiga dimensi.

Ok ... Tunggu apa lagi?? Kita telah selesai merakitnya, yuk kita coba!!!

Pelajarilah program berikut ini.

```
{-------------------------------------------     }
{ Program     : B3TRANS.PAS                     }
{ Coder       : Agustinus Nalwan                }
{ Description : Program untuk mencoba fungsi    }
{               transformasi pada MATH.TPU      }
{-------------------------------------------     }
Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Line_Rec = Record
       A,B : Integer;
     End;
```

```
    Vertex_Rec = Record
      X, Y, Z : Real;
    End;

    ObjRect = Record
      Color      : Byte;
      LineBank   : Array[0..20] of Line_Rec;
      VertexBank : Array[0..20] of Vertex_Rec;
      TotalVertex: Integer;
      TotalLine  : Integer;
    End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=x-lx;Box[No].VertexBank[0].Y:=y+ly;
       Box[No].VertexBank[0].Z:=z-lz;
    Box[No].VertexBank[1].X:=x+lx;Box[No].VertexBank[1].Y:=y+ly;
       Box[No].VertexBank[1].Z:=z-lz;
    Box[No].VertexBank[2].X:=x+ lx;Box[No].VertexBank[2].Y:=y-ly;
       Box[No].VertexBank[2].Z:=z-lz;
    Box[No].VertexBank[3].X:=x-lx;Box[No].VertexBank[3].Y:=y-ly;
       Box[No].VertexBank[3].Z:=z-lz;
    Box[No].VertexBank[4].X:=x-lx;Box[No].VertexBank[4].Y:=y+ly;
       Box[No].VertexBank[4].Z:=z+lz;
    Box[No].VertexBank[5].X:=x+lx;Box[No].VertexBank[5].Y:=y+ly;
       Box[No].VertexBank[5].Z:=z+lz;
    Box[No].VertexBank[6].X:=x+lx;Box[No].VertexBank[6].Y:=y-ly;
       Box[No].VertexBank[6].Z:=z+lz;
    Box[No].VertexBank[7].X:=x-lx;Box[No].VertexBank[7].Y:=y-ly;
       Box[No].VertexBank[7].Z:=z+lz;
    Box[No].LineBank[0].A:=0; Box[No].LineBank[0].B:=1;
    Box[No].LineBank[1].A:=1; Box[No].LineBank[1].B:=2;
    Box[No].LineBank[2].A:=2; Box[No].LineBank[2].B:=3;
    Box[No].LineBank[3].A:=3; Box[No].LineBank[3].B:=0;
    Box[No].LineBank[4].A:=4; Box[No].LineBank[4].B:=5;
    Box[No].LineBank[5].A:=5; Box[No].LineBank[5].B:=6;
    Box[No].LineBank[6].A:=6; Box[No].LineBank[6].B:=7;
    Box[No].LineBank[7].A:=7; Box[No].LineBank[7].B:=4;
    Box[No].LineBank[8].A:=0; Box[No].LineBank[8].B:=4;
    Box[No].LineBank[9].A:=1; Box[No].LineBank[9].B:=5;
    Box[No].LineBank[10].A:=2; Box[No].LineBank[10].B:=6;
    Box[No].LineBank[11].A:=3; Box[No].LineBank[11].B:=7;
    Box[No].TotalVertex:=8;
    Box[No].TotalLine:=12;
```

```
    Box[No].Color:=Col;
  End;

Procedure Initialize;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure Wire_Frame_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalLine-1 do
          Begin
           Psv_Projection(VertexBank[LineBank[A].A].X,
            VertexBank[LineBank[A].A].Y,
              VertexBank[LineBank[A].A].Z,X1,Y1);
           Psv_Projection(VertexBank[LineBank[A].B].X,
            VertexBank[LineBank[A].B].Y,
              VertexBank[LineBank[A].B].Z,X2,Y2);
           Line(Buff^,X1,Y1,X2,Y2,Color);
          End;
      End;
  End;

Procedure Display_Objek(No:Integer);
  Begin
    Wire_Frame_Projection(No);
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;
```

```
Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var TMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Dilatasi(TMat,Tx,Ty,Tz);
    For A:=0 to 2 do
      Transform(A,TMat);
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Display_Objek(0);
        Display_Objek(1);
        Display_Objek(2);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
```

```
      Case ch of
        'q':X_Rotation(2);
        'a':X_Rotation(-2);
        'w':Y_Rotation(2);
        's':Y_Rotation(-2);
        'e':Z_Rotation(2);
        'd':Z_Rotation(-2);
        'r':Scale(1.1,1,1);
        'f':Scale(0.9,1,1);
        't':Scale(1,1.1,1);
        'g':Scale(1,0.9,1);
        'y':Scale(1,1,1.1);
        'h':Scale(1,1,0.9);
        '7':Dilatasi(10,0,0);
        '1':Dilatasi(-10,0,0);
        '8':Dilatasi(0,10,0);
        '2':Dilatasi(0,-10,0);
        '9':Dilatasi(0,0,10);
        '3':Dilatasi(0,0,-10);
      End;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

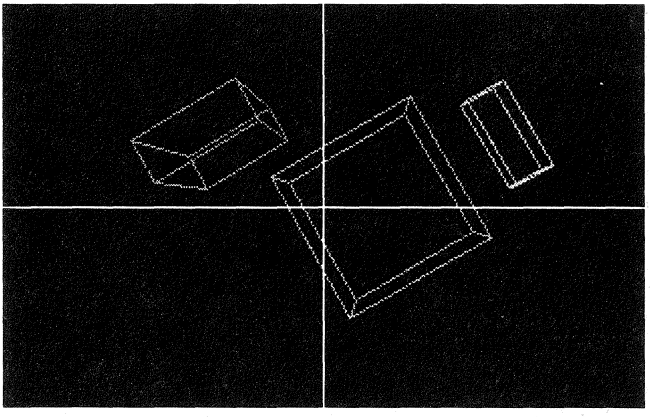
Pada program kita kali ini, kita punya banyak kebebasan. Gunakan tombol-tombol fungsi berikut ini:

Tombol Q	Rotasi sumbu X searah jarum jam.
Tombol A	Rotasi sumbu X berlawanan arah jarum jam.
Tombol W	Rotasi sumbu Y searah jarum jam.
Tombol S	Rotasi sumbu Y berlawanan arah jarum jam.
Tombol E	Rotasi sumbu Z searah jarum jam.
Tombol D	Rotasi sumbu Z berlawanan arah jarum jam.
Tombol R	Pembesaran terhadap sumbu X.
Tombol F	Pengecilan terhadap sumbu X.
Tombol T	Pembesaran terhadap sumbu Y.
Tombol G	Pengecilan terhadap sumbu Y.
Tombol Y	Pembesaran terhadap sumbu Z.
Tombol H	Pengecilan terhadap sumbu Z.
Tombol 7	Dilatasi ke sumbu X positif.
Tombol 1	Dilatasi ke sumbu X negatif.
Tombol 8	Dilatasi ke sumbu Y positif.

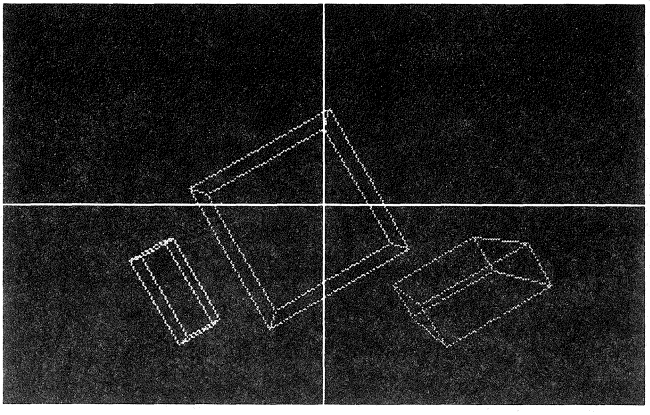
Tombol 2	Dilatasi ke sumbu Y negatif.
Tombol 9	Dilatasi ke sumbu Z positif.
Tombol 3	Dilatasi ke sumbu Z negatif.
ESC	Keluar.

Cobalah bermain-main sebentar dengan program ini.

WOW!!! Fungsi-fungsi transformasi yang telah kita rakit berjalan dengan mulus.



Objek setelah digeser dan dirotasi.



Objek yang ditransformasi lebih lanjut.

PENJELASAN LISTING PROGRAM

Pada dasarnya program ini hampir sama dengan program kita yang terdahulu yaitu B3VECT.PAS. Tetapi ada tambahan fungsi-fungsi transformasi sehingga memungkinkan kita memanipulasi objek.

Procedure ini berguna untuk melakukan suatu fungsi transformasi yang disimpan di dalam sebuah matrix (Matrix A) pada seluruh titik yang terletak pada VertexBank dari objek (Objek[No]) yang hendak ditransformasikan.

```
Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
```

Lakukan loop untuk mengakses seluruh titik di dalam Vertex Bank.

```
        For d:=0 to TotalVertex-1 do
          Begin
```

Masukkan koordinat titik yang sedang diakses ke dalam matrix 1 x 4.

```
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            Cmat[3]:=1;
```

Kalikan matrix koordinat Cmat dengan matrix operasi A sebagai proses transformasi.

```
            Mat_Multiply4x1(A,CMat);
```

Kembalikan koordinat dari Cmat ke Vertex Bank. Koordinat yang terdapat di dalam Cmat ini telah tertransformasi pada saat dilakukan perkalian matrix.

```
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;
```

Procedure X_Rotation berguna untuk mewujudkan rotasi terhadap sumbu X pada seluruh objek yang ada pada dunia tiga dimensi.

```
Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
```

Set matrix Xmat sebagai matrix rotasi terhadap sumbu X sebesar Deg derajat.

```
    Mat_XRotate(XMat,Deg);
```

Lakukan loop untuk men-transform seluruh objek.

```
    For A:=0 to 2 do
      Transform(A,XMat);
  End;
```

Procedure Y_Rotation berguna untuk mewujudkan rotasi terhadap sumbu Y pada seluruh objek yang ada pada dunia tiga dimensi.

```
Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
```

Set Matrix Ymat sebagai matrix rotasi terhadap sumbu Y sebesar Deg derajat.

```
    Mat_YRotate(YMat,Deg);
```

Lakukan loop untuk men-transform seluruh objek.

```
    For A:=0 to 2 do
      Transform(A,YMat);
  End;
```

Procedure Z_Rotation berguna untuk mewujudkan rotasi terhadap sumbu Z pada seluruh objek yang ada pada dunia tiga dimensi.

```
Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
```

Set Matrix Zmat sebagai matrix rotasi terhadap sumbu Z sebesar Deg dejarat.

```
    Mat_ZRotate(ZMat,Deg);
```

Lakukan loop untuk men-transform seluruh objek.

```
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;
```

Procedure ini bertujuan untuk mewujudkan fasilitas Scaling pada seluruh objek.

```
Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
```

Set matrix Smat sebagai matrix scalling terhadap sumbu X, Y dan Z sebesar Sx, Sy dand Sz.

```
    Mat_Scaling(SMat,Sx,Sy,Sz);
```

Lakukan loop untuk men-transform seluruh objek.

```
    For A:=0 to 2 do
      Transform(A,SMat);
  End;
```

Procedure ini berguna untuk mewujudkan fasilitas dilatasi terhadap seluruh objek.

```
Procedure Dilatasi(Tx,Ty,Tz:Real);
Var TMat:Mat4x4;
    A:Integer;
  Begin
```

Set matrix Tmat sebagai matrix dilatasi terhadap sumbu X, Y dan Z sebesar Tx, Ty dan Tz.

```
    Mat_Dilatasi(TMat,Tx,Ty,Tz);
```

Lakukan loop untuk men-transform seluruh objek.

```
    For A:=0 to 2 do
      Transform(A,TMat);
  End;
```

Pada main program, tidak dilakukan banyak perubahan. Hanya diberikan tambahan pengoperasian keyboard untuk melakukan rotasi, dilatasi dan scaling.

```
Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Display_Objek(0);
        Display_Objek(1);
        Display_Objek(2);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
      Case ch of
```

Lakukan rotasi sebesar dua derajat.

```
        'q':X_Rotation(2);
        'a':X_Rotation(-2);
        'w':Y_Rotation(2);
        's':Y_Rotation(-2);
        'e':Z_Rotation(2);
        'd':Z_Rotation(-2);
```

Lakukan scalling 110 persen terhadap sumbu X pada objek, 100 persen (ukuran asli) pada sumbu Y dan 100 persen (ukuran asli) pada sumbu Z. Sehingga koordinat X objek dibesarkan 1.1 kali, tetapi koordinat Y dan Z tetap tidak berubah karena pembesaran dilakukan 100 persen = 1 kali.

```
        'r':Scale(1.1,1,1);
```

Lakukan unscalling 90 persen terhadap sumbu X pada objek, 100 persen (ukuran asli) pada sumbu Y dan Z. Sehingga koordinat X objek dikecilkan 0.9 kali.

```
        'f':Scale(0.9,1,1);
```

Lakukan scalling 110 persen terhadap sumbu Y pada objek, 100 persen (ukuran asli) pada sumbu X dan 100 persen (ukuran asli) pada sumbu Z. Sehingga koordinat Y objek dibesarkan 1.1 kali, tetapi koordinat X dan Z tetap tidak berubah karena pembesaran dilakukan 100 persen = 1 kali.

```
        't':Scale(1,1.1,1);
```

Selebihnya, proses yang sama dilakukan lagi.

```
        'g':Scale(1,0.9,1);
        'y':Scale(1,1,1.1);
        'h':Scale(1,1,0.9);
```

Lakukan dilatasi sejauh 10 terhadap koordinat X objek.

```
        '7':Dilatasi(10,0,0);
```

Lakukan dilatasi sejauh -10 terhadap koordinat X objek.

```
        '1':Dilatasi(-10,0,0);
```

Lakukan dilatasi sejauh 10 terhadap koordinat Y objek.

```
        '8':Dilatasi(0,10,0);
```

Selebihnya, proses yang sama diulang lagi.

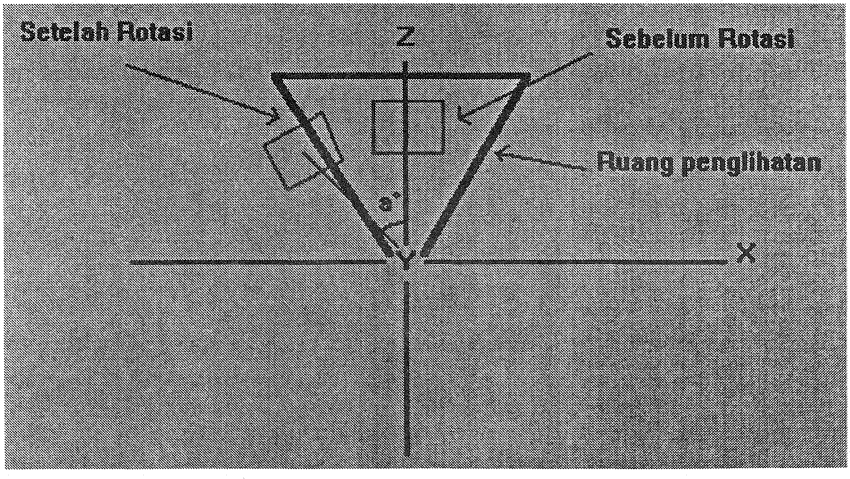
```
        '2':Dilatasi(0,-10,0);
        '9':Dilatasi(0,0,10);
        '3':Dilatasi(0,0,-10);
      End;
    Until Ch=#27;
  End;
```

BAB 4

LOCAL vs GLOBAL COORDINATE SYSTEM TRANSFORMATION

4.1 Local dan Global Coordinate System

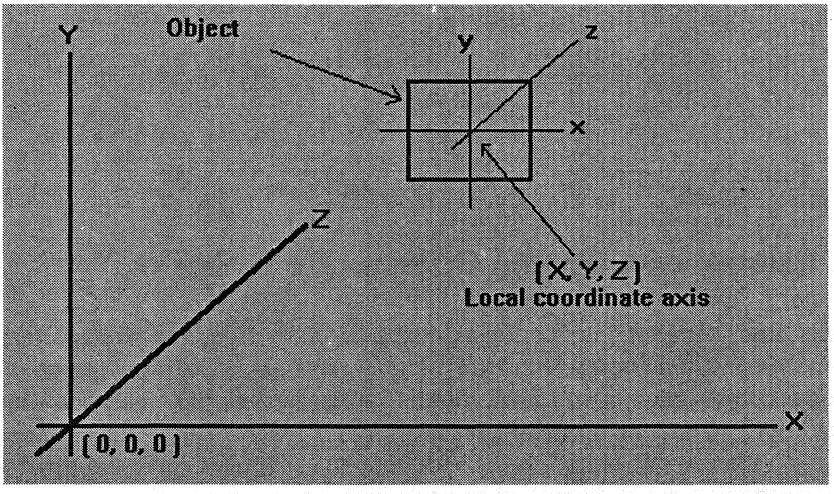
Pada bab terdahulu, telah diulas habis-habisan mengenai transformasi. Sebenarnya teknik transformasi yang kita gunakan tersebut masih tergolong global coordinate system transformation. Artinya ialah, titik-titik pada objek kita transformasikan berdasarkan sumbu X, Y dan Z dunia tiga dimensi. Rotasi sumbu Y terhadap objek dengan global coordinate system transformation dapat digambarkan pada diagram di bawah ini.



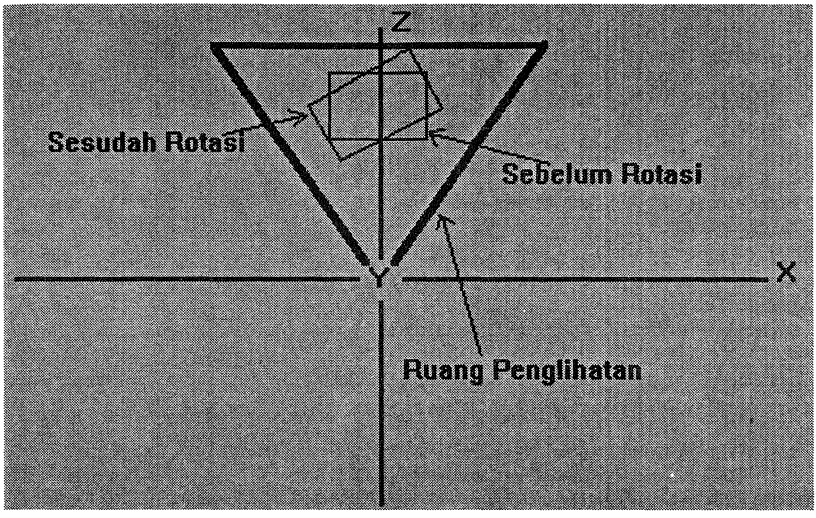
Perhatikan objek sebelum dan setelah dirotasi. Objek tersebut bergeser karena dilakukannya rotasi terhadap sumbu Y. Global coordinate system transformation ini merotasikan objek terhadap sumbu Y dunia (sumbu Y global).

Sehingga pada layar monitor, objek terlihat hanya seperti bergeser ke arah kiri. Padahal sebenarnya objek itu berputar hampir mengelilingi kita.

Pada dunia tiga dimensi dikenal sebuah teknik transformasi yang lain yaitu local coordinate system transformation. Di sini objek dirotasi terhadap titik pivot/titik putarnya. Pada titik pivot setiap objek terdapat koordinat axis lokal.



Perhatikan diagram di atas. Local coordinate axis dari objek di atas ialah pada titik tengah dari objek tersebut. Pada titik pivot objek tersebut terdapat sumbu X, Y dan Z sendiri. Untuk melengkapi teknik ini, koordinat setiap titik pada Vertex Bank ialah koordinat lokal dari objek bukannya koordinat dunia seperti apa yang telah kita pakai sebelumnya.



Perhatikan diagram di atas. Setelah dirotasi terhadap local coordinat Y axisnya, objek tersebut tidak berpindah tempat. Cara ini digunakan oleh game-game flight simulator untuk memvisualisasikan gerakan pesawat. Jika pesawat hendak berputar ke kiri/ke kanan, sebenarnya pesawat tersebut diputar terhadap sumbu Y local coordinat axisnya bukannya terhadap sumbu Y dunia.

Untuk lebih jelasnya lihat program di bawah ini.

```
{-----------------------------------------------}
{ Program     : B3LOC.PAS                       }
{ Coder       : Agustinus Nalwan                }
{ Description: Program transformasi objek       }
{              dengan teknik local coordinate   }
{              system transformation            }
{-----------------------------------------------}
Uses Math,Crt,Dos,VgaCad;
Const View_Distance = 400;

Type
     Line_Rec = Record
       A,B : Integer;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
     End;

     ObjRect = Record
       Color       : Byte;
```

```
      LineBank   : Array[0..20] of Line_Rec;
      VertexBank : Array[0..20] of Vertex_Rec;
      TotalVertex: Integer;
      TotalLine  : Integer;
      X,Y,Z      : Real;
    End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
    Box[No].VertexBank[0].X:=-lx;
        Box[No].VertexBank[0].Y:=ly;Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;Box[No].VertexBank[1].Y:=ly;
        Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
        Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
        Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;
        Box[No].VertexBank[4].Y:=ly;Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
        Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
        Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
        Box[No].VertexBank[7].Z:=lz;
    Box[No].LineBank[0].A:=0; Box[No].LineBank[0].B:=1;
    Box[No].LineBank[1].A:=1; Box[No].LineBank[1].B:=2;
    Box[No].LineBank[2].A:=2; Box[No].LineBank[2].B:=3;
    Box[No].LineBank[3].A:=3; Box[No].LineBank[3].B:=0;
    Box[No].LineBank[4].A:=4; Box[No].LineBank[4].B:=5;
    Box[No].LineBank[5].A:=5; Box[No].LineBank[5].B:=6;
    Box[No].LineBank[6].A:=6; Box[No].LineBank[6].B:=7;
    Box[No].LineBank[7].A:=7; Box[No].LineBank[7].B:=4;
    Box[No].LineBank[8].A:=0; Box[No].LineBank[8].B:=4;
    Box[No].LineBank[9].A:=1; Box[No].LineBank[9].B:=5;
    Box[No].LineBank[10].A:=2; Box[No].LineBank[10].B:=6;
    Box[No].LineBank[11].A:=3; Box[No].LineBank[11].B:=7;
    Box[No].TotalVertex:=8;
    Box[No].TotalLine:=12;
    Box[No].Color:=Col;
  End;

Procedure Initialize;
```

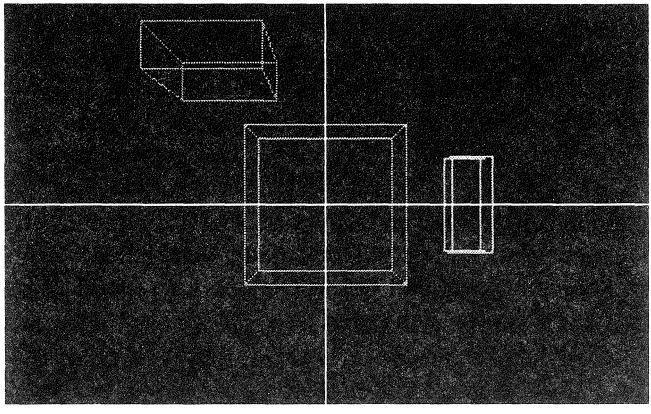
```
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure Wire_Frame_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalLine-1 do
          Begin
           Psv_Projection(VertexBank[LineBank[A].A].X+X,
               VertexBank[LineBank[A].A].Y+Y,
              VertexBank[LineBank[A].A].Z+Z,X1,Y1);
            Psv_Projection(VertexBank[LineBank[A].B].X+X,
               VertexBank[LineBank[A].B].Y+Y,
              VertexBank[LineBank[A].B].Z+Z,X2,Y2);
            Line(Buff^,X1,Y1,X2,Y2,Color);
          End;
      End;
  End;

Procedure Display_Object(No:Integer);
  Begin
    Wire_Frame_Projection(No);
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
```

```
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var TMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Dilatasi(TMat,Tx,Ty,Tz);
    For A:=0 to 2 do
      Transform(A,TMat);
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Display_Objek(0);
        Display_Objek(1);
        Display_Objek(2);
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
      Case ch of
        'q':X_Rotation(2);
        'a':X_Rotation(-2);
```

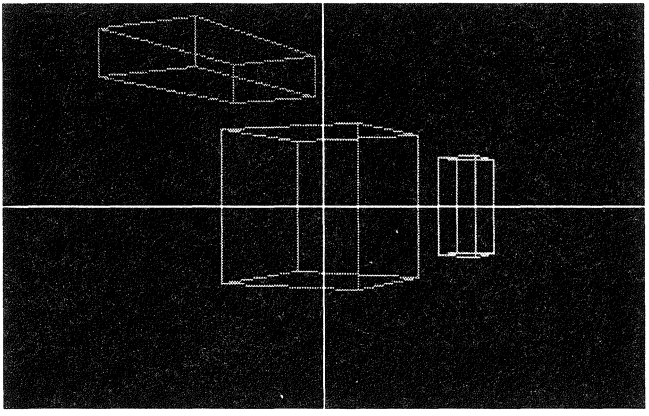
```
        'w':Y_Rotation(2);
        's':Y_Rotation(-2);
        'e':Z_Rotation(2);
        'd':Z_Rotation(-2);
        'r':Scale(1.1,1,1);
        'f':Scale(0.9,1,1);
        't':Scale(1,1.1,1);
        'g':Scale(1,0.9,1);
        'y':Scale(1,1,1.1);
        'h':Scale(1,1,0.9);
        '7':Dilatasi(10,0,0);
        '1':Dilatasi(-10,0,0);
        '8':Dilatasi(0,10,0);
        '2':Dilatasi(0,-10,0);
        '9':Dilatasi(0,0,10);
        '3':Dilatasi(0,0,-10);
      End;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Coba jalankan program di atas. Sekarang cobalah untuk merotasi objek. Kelihatan perbedaannya?? Sekarang objek dirotasi berdasarkan sumbu lokal koordinatnya sendiri-sendiri.



Objek sebelum dirotasi terhadap koordinat sistem lokalnya.



Objek setelah dirotasi terhadap sumbu Y koordinat sistem lokalnya. Perhatikan bahwa posisi objek tidak ikut bergeser.

PENJELASAN LISTING PROGRAM

Sebenarnya penggantian teknik proyeksi ini tidak membutuhkan banyak perubahan yang menyolok. Perubahan pertama dilakukan terhadap struktur data dari objek. Di sini ditambahkan X, Y dan Z yang berfungsi untuk menyimpan koordinat titik pivot dari objek. Pada titik pivot inilah terletak koordinat sistem lokal dari objek.

```
ObjRect = Record
  Color      : Byte;
  LineBank   : Array[0..20] of Line_Rec;
  VertexBank : Array[0..20] of Vertex_Rec;
  TotalVertex: Integer;
  TotalLine  : Integer;
  X,Y,Z      : Real;
End;
```

Perubahan kedua yang dilakukan ialah pada saat membentuk objek. Seperti yang telah kita pelajari bahwa titik-titik pada Vertex Bank akan disimpan berdasarkan koordinat lokal dari objek. Dengan kata lain, koordinat (0,0,0) lokal terletak pada titik pivot dari objek.

```
Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
```

Tentukan koordinat titik pivot dari objek.

```
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
```

Masukkan koordinat lokal dari tiap titik. Perhatikan, di sini tidak lagi dilakukan penambahan terhadap titik pusat dari objek seperti yang biasa kita lakukan. Hal ini disebabkan karena titik tengah dari objek inilah yang kita

pakai sebagai pivot. Dan karena koordinat lokal dari objek adalah relative terhadap titik pivot, maka kita tidak perlu menambahkan koordinat titik pivot tersebut.

```
    Box[No].VertexBank[0].X:=-lx;
        Box[No].VertexBank[0].Y:=ly;Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;
        Box[No].VertexBank[1].Y:=ly;Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
        Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
        Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;
        Box[No].VertexBank[4].Y:=ly;Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
        Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
        Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
        Box[No].VertexBank[7].Z:=lz;
    Box[No].LineBank[0].A:=0; Box[No].LineBank[0].B:=1;
    Box[No].LineBank[1].A:=1; Box[No].LineBank[1].B:=2;
    Box[No].LineBank[2].A:=2; Box[No].LineBank[2].B:=3;
    Box[No].LineBank[3].A:=3; Box[No].LineBank[3].B:=0;
    Box[No].LineBank[4].A:=4; Box[No].LineBank[4].B:=5;
    Box[No].LineBank[5].A:=5; Box[No].LineBank[5].B:=6;
    Box[No].LineBank[6].A:=6; Box[No].LineBank[6].B:=7;
    Box[No].LineBank[7].A:=7; Box[No].LineBank[7].B:=4;
    Box[No].LineBank[8].A:=0; Box[No].LineBank[8].B:=4;
    Box[No].LineBank[9].A:=1; Box[No].LineBank[9].B:=5;
    Box[No].LineBank[10].A:=2; Box[No].LineBank[10].B:=6;
    Box[No].LineBank[11].A:=3; Box[No].LineBank[11].B:=7;
    Box[No].TotalVertex:=8;
    Box[No].TotalLine:=12;
    Box[No].Color:=Col;
  End;
```

Perubahan terakhir ialah pada saat dilakukan proyeksi. Mengingat data koordinat pada Vertex Bank adalah koordinat lokal. Untuk melakukan proyeksi, diperlukan koordinat global dari titik tersebut. Untuk itu dilakukan penambahan dengan koordinat titik pivot pada setiap titik yang hendak diproyeksikan.

```
Procedure Wire_Frame_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
```

```
    Begin
      For A:=0 to TotalLine-1 do
        Begin
```

Dilakukan penambahan dengan koordinat titik pivot yaitu X, Y dan Z.

```
          Psv_Projection(VertexBank[LineBank[A].A].X+X,
            VertexBank[LineBank[A].A].Y+Y,
              VertexBank[LineBank[A].A].Z+Z,X1,Y1);
            Psv_Projection(VertexBank[LineBank[A].B].X+X,
              VertexBank[LineBank[A].B].Y+Y,
              VertexBank[LineBank[A].B].Z+Z,X2,Y2);
            Line(Buff^,X1,Y1,X2,Y2,Color);
          End;
    End;
End;
```

BAB 5

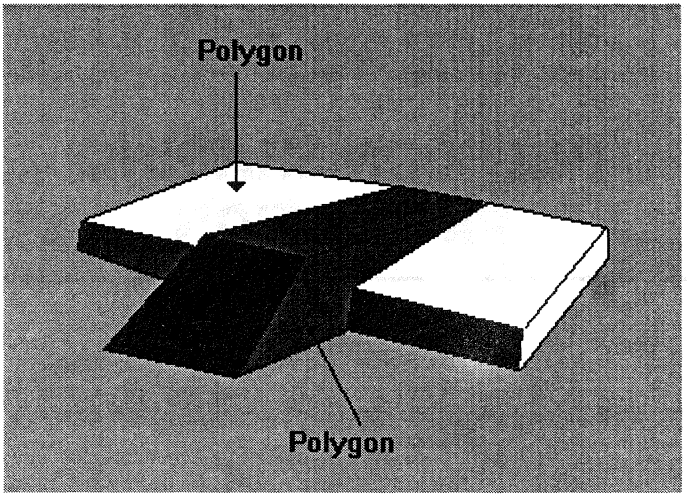
POLYGON SHADING PROJECTION

5.1 Polygon Shading Data Structure

Poligon ialah bangun/kurva tertutup yang dibentuk dari sejumlah titik yang dihubungkan dengan garis.

Dengan ditemukannya teknik wireframe projection, para game-game developer masih belum juga puas. Karena hasil dari wireframe projection ini bisa dibilang jauh dari realistis. Objek terlihat transparan sehingga sulit dikenali. Dengan semakin cepatnya perkembangan processor komputer yaitu 286 dan 386 pada saat itu, dikenal teknik baru yang disebut polygon shading. Teknik ini tidak mungkin dilakukan dengan komputer XT mengingat keterbatasan register yang ada (8 bit) dan juga kecepatan CPU per instruction yang sangat lambat.

Teknik Polygon Shading ini dilakukan dengan mengarsir polygon hasil proyeksi dengan warna sehingga objek hasil proyeksi terlihat lebih realistis dan tidak terlihat transparan seperti pada wireframe projection.



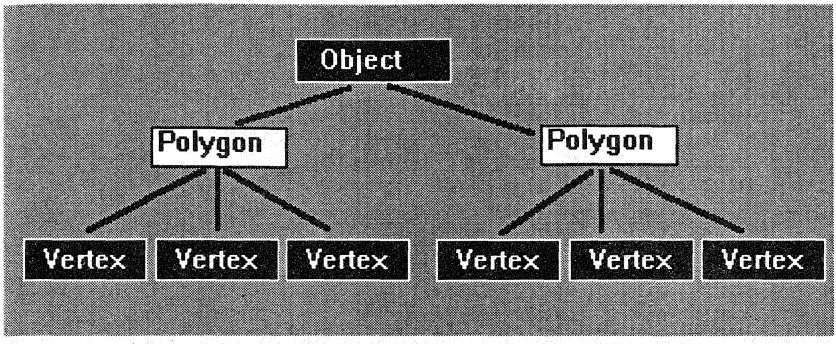
Inilah teknik Polygon Shading. Objek terlihat lebih realistis. Game-game yang mendukung teknik ini ialah F-19, F-15, Test Drive III dan sebangsanya. Polygon shading technique ini dilakukan pada saat seluruh titik telah diproyeksikan ke layar monitor. Untuk itu kita perlu melakukan sedikit perubahan pada objek data structure yang kita ciptakan pada saat kita membahas wireframe projection. Mengapa? Karena pada wireframe projection kita tidak peduli akan keberadaan polygon/face melainkan cukup memproyeksikan segment/garis saja.

Di sini kita perlu menyimpan informasi poligon sampai pada saat proyeksi karena informasi itu diperlukan untuk menentukan warna poligon yang harus diarsir.

Jenis data structure yang akan kita bahas ada dua macam. Tetapi hanya satu yang akan kita pakai sebagai data structure standard pada pembahasan buku ini.

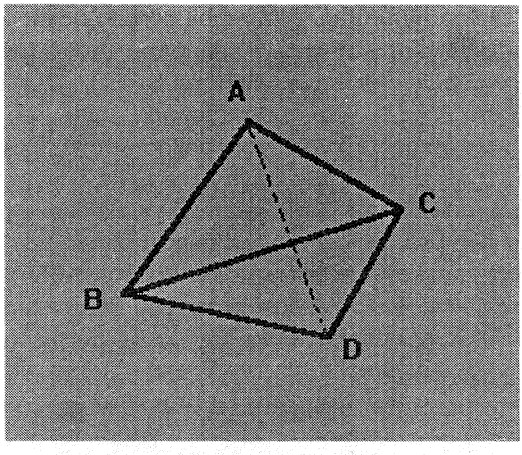
Jenis data structure yang pertama ialah sebagai berikut:

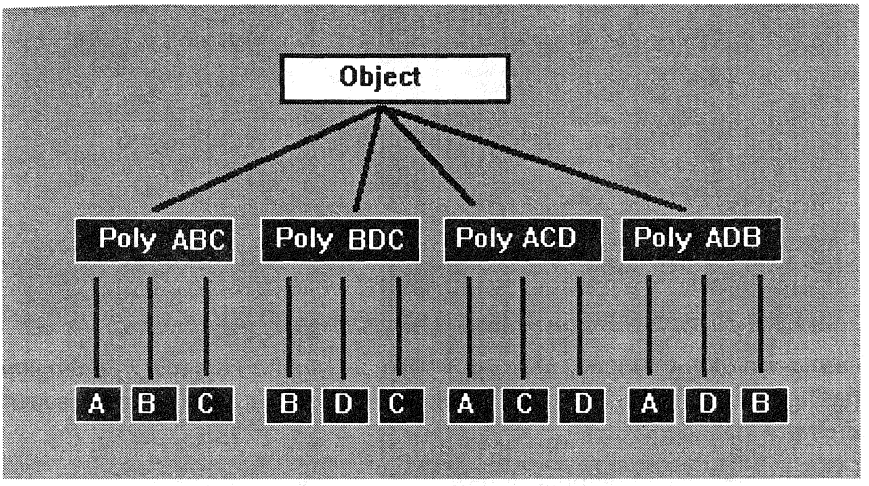
Objek ialah kumpulan/array of Polygon sedangkan masing-masing Poligon ialah kumpulan/array of Vertex.



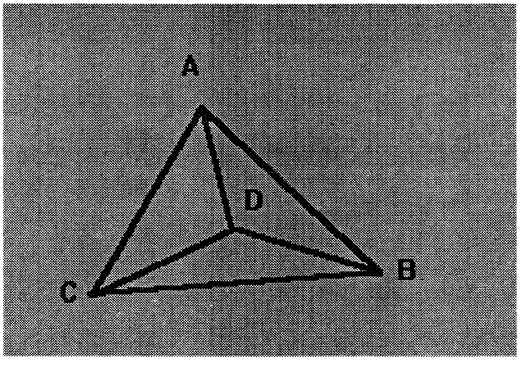
Masing-masing Poligon memiliki informasi warna arsiran. Sedangkan masing-masing Vertex memiliki koordinat tiga dimensi (sama seperti wireframe data structure).

Sekarang perhatikan gambar di bawah ini, di mana ditunjukkan bagaimana cara menyimpan sebuah objek dalam sistem data structure ini.





Inilah Data Structure yang dibentuk untuk menyimpan objek pada gambar di atas. Dengan kata lain, kita harus menyimpan setiap titik pada masing-masing poligon secara berurutan. Kita ambil contoh poligon ABC, dibentuk oleh garis AB, BC, dan CA sesuai dengan urutan penyimpanannya. Kemudian poligon BDC dibentuk oleh garis BD, DC dan CB. Perlu diperhatikan dalam melakukan penyimpanan titik, dilakukan secara berlawanan arah jarum jam. Perhatikan poligon ABC, dilakukan penyimpanan mulai dari A ke B ke C bukannya A ke C ke B karena A ke B ke C adalah berlawanan arah jarum jam. Sekarang perhatikan poligon ACD, jika diperhatikan dari gambar, kelihatan arah penyimpanan titik dari poligon ini searah jarum jam, tetapi sebenarnya tidak. Poligon ACD pada gambar di atas membelakangi kita, jika objek kita putar 180 derajat terhadap sumbu Y maka segitiga ACD akan menghadap ke arah kita seperti gambar berikut ini.

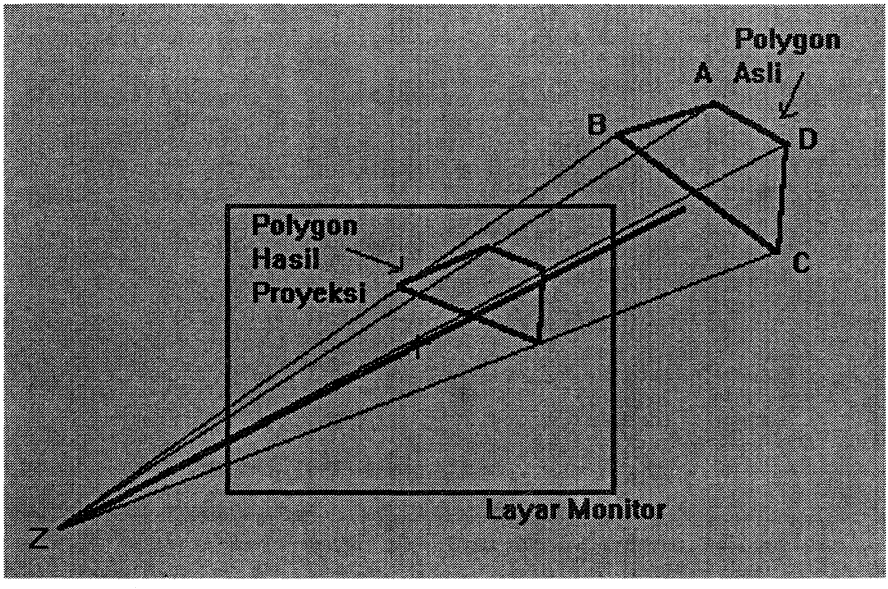


Nah di sini baru terlihat bahwa penyimpanan mulai dari A ke C ke D sebenarnya berlawanan arah jarum jam. Tujuan dari penyimpanan secara

berlawanan arah jarum jam ini berkaitan dengan BACKFACE REMOVAL technique yang akan segera dibahas.

Teknik inilah yang akan kita pergunakan dalam pembahasan tiga dimensi pada buku ini karena mudah dimengerti.

Masalah kita tidak berakhir hanya sampai di sini. Kita harus mampu memproyeksikan objek yang disimpan dengan data structure ini dengan menggunakan teknik polygon shading. Mula-mula lakukan loop untuk mengakses setiap poligon dalam sebuah objek. Kemudian untuk masing-masing poligon kita lakukan sebuah loop lagi untuk memproyeksikan seluruh titik yang membentuk poligon tersebut. Berikutnya, setelah kita memproyeksikan seluruh titik dari poligon, masukkan poligon tersebut ke dalam sebuah polygon list (polygon bank). Polygon list ini berguna untuk menyimpan seluruh polygon hasil proyeksi. Karena yang disimpan adalah polygon hasil proyeksi, maka masing-masing elemen dari polygon list adalah array of Vertex di mana masing-masing Vertex hanya berisi informasi koordinat dua dimensi (X dan Y).



Sebelum diproyeksi, poligon dari sebuah objek masih terdiri dari Vertex yang memiliki koordinat tiga dimensi (X, Y, Z) tetapi setelah diproyeksikan ke layar monitor, poligon tersebut hanya terdiri dari Vertex dua dimensi (X, Y).

Nah, setelah poligon yang telah diproyeksikan beserta informasi warna arsiran dimasukkan ke dalam Polygon List, kita lakukan proyeksi terhadap

poligon berikutnya hingga seluruh poligon dari objek telah terproyeksi.

Di bawah ini adalah program B3TRANS.PAS kita yang terdahulu setelah dimodifikasi sehingga menggunakan polygon shading data structure.

```
{-----------------------------------------------        }
{ Program     : B3PLIST.PAS                            }
{ Coder       : Agustinus Nalwan                       }
{ Description : Program proyeksi wireframe dengan      }
{               mempergunakan polygon shading data     }
{               structure.                             }
{-----------------------------------------------        }

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
       Xp,Yp   : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
     End;

     PolyList_Rec = Record
       ObjNo  : Integer;
       PolyNo : Integer;
       Active : Boolean;
     End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ListCtr:Integer;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
```

```
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=x-lx;
        Box[No].VertexBank[0].Y:=y+ly;Box[No].VertexBank[0].Z:=z-lz;
    Box[No].VertexBank[1].X:=x+lx;Box[No].VertexBank[1].Y:=y+ly;
        Box[No].VertexBank[1].Z:=z-lz;
    Box[No].VertexBank[2].X:=x+lx;Box[No].VertexBank[2].Y:=y-ly;
        Box[No].VertexBank[2].Z:=z-lz;
    Box[No].VertexBank[3].X:=x-lx;Box[No].VertexBank[3].Y:=y-ly;
        Box[No].VertexBank[3].Z:=z-lz;
    Box[No].VertexBank[4].X:=x-lx;
        Box[No].VertexBank[4].Y:=y+ly;Box[No].VertexBank[4].Z:=z+lz;
    Box[No].VertexBank[5].X:=x+lx;Box[No].VertexBank[5].Y:=y+ly;
        Box[No].VertexBank[5].Z:=z+lz;
    Box[No].VertexBank[6].X:=x+lx;Box[No].VertexBank[6].Y:=y-ly;
        Box[No].VertexBank[6].Z:=z+lz;
    Box[No].VertexBank[7].X:=x-lx;Box[No].VertexBank[7].Y:=y-ly;
        Box[No].VertexBank[7].Z:=z+lz;
    Box[No].Polygon[0].TotalVertex:=4;
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;

    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;
```

```
    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].Color:=Col;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
    Remove_PolyList;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X,VertexBank[A].Y,
              VertexBank[A].Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          For B:=0 to Polygon[A].TotalVertex do
            Begin
              PolygonList[ListCtr].ObjNo:=No;
              PolygonList[ListCtr].PolyNo:=A;
              PolygonList[ListCtr].Active:=True;
              Inc(ListCtr);
            End;
      End;
  End;

Procedure Display_PList_WF;
```

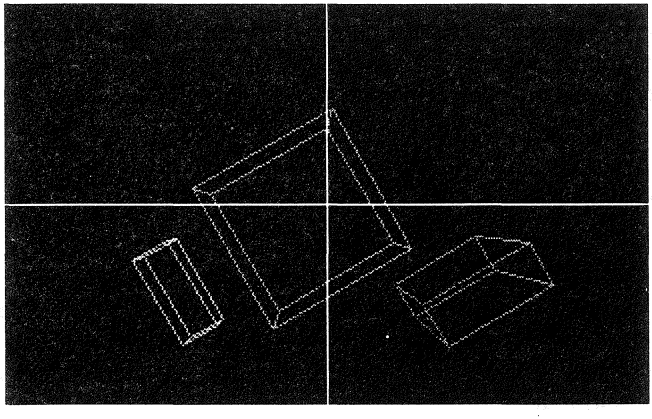
```
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;            Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
```

```
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var TMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Dilatasi(TMat,Tx,Ty,Tz);
    For A:=0 to 2 do
      Transform(A,TMat);
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Objek;
        Display_PList_WF;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
      Case ch of
        'q':X_Rotation(2);
        'a':X_Rotation(-2);
        'w':Y_Rotation(2);
        's':Y_Rotation(-2);
```

```
        'e':Z_Rotation(2);
        'd':Z_Rotation(-2);
        'r':Scale(1.1,1,1);
        'f':Scale(0.9,1,1);
        't':Scale(1,1.1,1);
        'g':Scale(1,0.9,1);
        'y':Scale(1,1,1.1);
        'h':Scale(1,1,0.9);
        '7':Dilatasi(10,0,0);
        '1':Dilatasi(-10,0,0);
        '8':Dilatasi(0,10,0);
        '2':Dilatasi(0,-10,0);
        '9':Dilatasi(0,0,10);
        '3':Dilatasi(0,0,-10);
      End;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Dilihat dari segi praktis, program ini memiliki output yang sama persis dengan program B3TRANS.PAS, juga tombol-tombol fungsi yang digunakan untuk mentransformasi objek. Gerakkan objek ke arah sumbu X, Y, Z juga lakukan rotasi sumbu X, Y dan Z. Tidak ada perubahan ... Lantas apanya yang berubah?



Di sini objek digambar per-poligon bukanya per-line seperti yang sebelumnya kita lakukan.

PENJELASAN LISTING PROGRAM

Secara teknis, struktur data yang kita gunakan pada program ini sangat berbeda dengan sistem data yang kita pergunakan pada program B3TRANS.PAS pada bab yang terdahulu. Hal ini disebabkan oleh penggambaran yang dilakukan di sini dilakukan secara per-poligon, bukannya per-garis seperti pada B3TRANS.PAS. Untuk itu digunakan polygon list yang berguna untuk menampung seluruh poligon yang hendak digambar pada layar monitor. Penggunaan polygon list ini sangatlah penting dalam pemrograman tiga dimensi dan juga merupakan keharusan bagi setiap program tiga dimensi yang memanfaatkan vektor. Dari polygon list inilah filter-filter untuk meminimumkan jumlah poligon yang harus digambar dilakukan. Penjelasan mengenai filter-filter ini akan diberikan segera.

Ok, kita mulai saja dengan pembahasan listing program.

Poly_Rec ialah Polygon Bank yang berfungsi untuk menyimpan seluruh poligon dalam sebuah objek. Adapun jumlah titik yang menghubungkan poligon ini dapat berjumlah 11. Informasi titik ini berupa Integer yang berisi index dari titik dalam Vertex Bank yang dipakai.

TotalVertex menunjukkan jumlah titik pada poligon tersebut.

```
Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
     End;
```

Pada elemen Vertex Bank ditambahkan Xp dan Yp yang berfungsi untuk menampung hasil proyeksi dari titik X, Y, Z. Dengan cara ini maka kita dapat melakukan proses proyeksi dan penggambaran poligon secara terpisah.

```
     Vertex_Rec = Record
       X, Y, Z : Real;
       Xp,Yp   : Longint;
     End;
```

Sebuah objek yang mengikuti format polygon shading data structure, memiliki dua buah elemen yang penting. Elemen pertama ialah Vertex Bank yaitu tempat disimpannya seluruh titik dari objek. Kedua ialah Polygon Bank (variabel poligon) yang berisi seluruh poligon dalam objek. Keduanya memiliki jumlah maksimum 21.

```
     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
```

```
    TotalPoly  : Integer;
  End;
```

Ini adalah data structure dari Polygon List yang akan kita gunakan. Polygon list berguna untuk menampung seluruh poligon hasil transformasi. Adapun secara praktis, Polygon List ini hanyalah index yang menunjuk ke sebuah poligon dari sebuah objek yang ditunjuk oleh variabel ObjNo dan PolyNo. Active ialah variabel yang menunjukkan apakah Polygon List elemen tersebut terpakai atau tidak terpakai/kosong.

```
  PolyList_Rec = Record
    ObjNo  : Integer;
    PolyNo : Integer;
    Active : Boolean;
  End;

Var A,B:Integer;
  Box:array[0..2] of ObjRect;
```

Deklarasikan Polygon List berjumlah 101. Dengan kata lain, jumlah Polygon List ini paling sedikit harus sama dengan jumlah seluruh poligon dari seluruh objek yang ada pada dunia tiga dimensi kita.

```
  PolygonList:array[0..100] of PolyList_Rec;
  ListCtr:Integer;
  buff:Pointer;
```

Kedua procedure di bawah ini digunakan untuk melakukan proyeksi dengan teknik yang berbeda. Adapun Pll_Projection tidak akan kita pakai lagi dalam pembahasan selanjutnya.

```
Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;
Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;
```

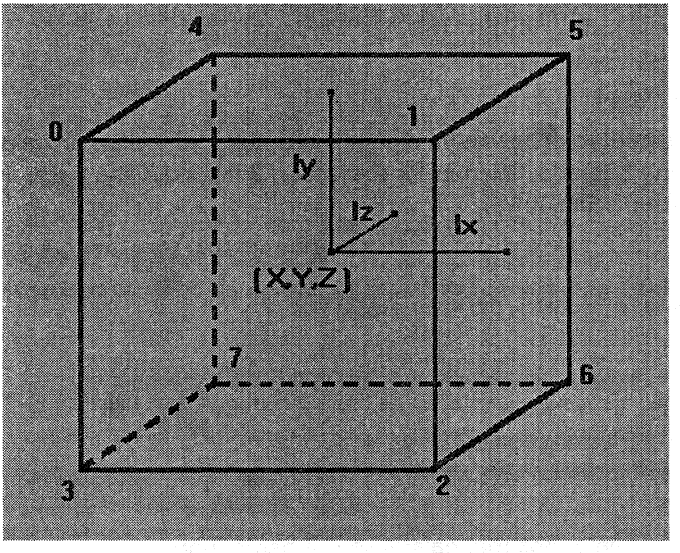
Procedure Create_Box mengalami sedikit perubahan.

```
Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
```

Alokasi titik pada VertexBank masih tetap sama seperti cara yang dulu. Jumlah titik pada balok berjumlah 8.

```
    Box[No].VertexBank[0].X:=x-lx;Box[No].VertexBank[0].Y:=y+ly;
      Box[No].VertexBank[0].Z:=z-lz;
```

```
Box[No].VertexBank[1].X:=x+lx;Box[No].VertexBank[1].Y:=y+ly;
    Box[No].VertexBank[1].Z:=z-lz;
Box[No].VertexBank[2].X:=x+lx;Box[No].VertexBank[2].Y:=y-ly;
    Box[No].VertexBank[2].Z:=z-lz;
Box[No].VertexBank[3].X:=x-lx;Box[No].VertexBank[3].Y:=y-ly;
    Box[No].VertexBank[3].Z:=z-lz;
Box[No].VertexBank[4].X:=x-lx;Box[No].VertexBank[4].Y:=y+ly;
    Box[No].VertexBank[4].Z:=z+lz;
Box[No].VertexBank[5].X:=x+lx;Box[No].VertexBank[5].Y:=y+ly;
    Box[No].VertexBank[5].Z:=z+lz;
Box[No].VertexBank[6].X:=x+lx;Box[No].VertexBank[6].Y:=y-ly;
    Box[No].VertexBank[6].Z:=z+lz;
Box[No].VertexBank[7].X:=x-lx;Box[No].VertexBank[7].Y:=y-ly;
    Box[No].VertexBank[7].Z:=z+lz;
Box[No].Polygon[0].TotalVertex:=4;
Box[No].Polygon[1].TotalVertex:=4;
Box[No].Polygon[2].TotalVertex:=4;
Box[No].Polygon[3].TotalVertex:=4;
Box[No].Polygon[4].TotalVertex:=4;
Box[No].Polygon[5].TotalVertex:=4;
```



Lakukan perancangan poligon satu demi satu. Poligon pertama yang berindex 0, dihubungkan oleh titik pada Vertex Bank nomor 0, 3, 2, 1 kemudian kembali ke 0. Titik terakhir selalu akan dihubungkan dengan titik pertama. Perhatikan gambar di atas. Urutan penyimpanan titik dilakukan berlawanan arah jarum jam (mulai titik sudut 0 ke 3 ke 2 ke 1 kembali ke 0).

```
    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;
```

Lakukan perancangan poligon sisi atas.

```
    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;
```

Lakukan perancangan poligon sisi belakang.

```
    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;
```

Lakukan perancangan poligon sisi bawah.

```
    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;
```

Lakukan perancangan poligon sisi kiri.

```
    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;
```

Lakukan perancangan poligon sisi kanan.

```
    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].Color:=Col;
End;
```

Procedure di bawah ini berguna untuk menginisialisasi Polygon List yang dipakai. Kosongkan seluruh element-nya dan set ListCtr yang berarti jumlah element Polygon List yang dipakai menjadi 0.

Proses ini harus dilakukan pada setiap awal loop untuk melakukan Polygon Assignment yang baru.

```
Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
```

Instal Grafik dan Virtual Screen.

```
    OpenVga256;
    GetMem(Buff,64000);
```

Ciptakan tiga buah objek pada dunia tiga dimensi.

```
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
```

Inisialisasikan Polygon List.

```
    Remove_PolyList;
  End;
```

Kembali ke mode text.

```
Procedure DeInitialize;
  Begin
    CloseVga256;
  End;
```

Pada program ini, proyeksi titik dilakukan terlebih dahulu dan hasil proyeksinya disimpan dalam Vertex Bank itu sendiri pada variabel Xp dan Yp. Cara ini digunakan untuk menyederhanakan masalah.

```
Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
```

Lakukan counter untuk seluruh titik pada Vertex Bank.

```
        For A:=0 to TotalVertex-1 do
          Begin
```

Proyeksikan titik tersebut. Masukkan hasil proyeksi ke dalam Xp dan Yp dalam Vertex Bank itu sendiri untuk keperluan lebih lanjut.

```
            Psv_Projection(VertexBank[A].X,VertexBank[A].Y,
              VertexBank[A].Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;
```

Di sini dilakukan Polygon List assignment yaitu memasukkan seluruh poligon yang ada di dunia tiga dimensi ke dalam Polygon List. Parameter yang dikirimkan ialah nomor objek yang poligon-poligonnya hendak dimasukkan ke dalam Polygon List.

```
Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
```

Lakukan counter untuk seluruh poligon dalam objek.

```
        For A:=0 to TotalPoly-1 do
          Begin
```

Masukkan nomor poligon dan nomor objek ke dalam Polygon List. Kemudian aktifkan Polygon List tersebut.

```
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
```

Tambah jumlah element PolygonList yang dipakai.

```
            Inc(ListCtr);
        End;
    End;
  End;
```

Procedure di bawah ini berguna untuk melakukan penggambaran dari seluruh poligon yang didaftar di dalam Polygon List secara wire frame.

```
Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
```

Lakukan counter untuk mengakses seluruh element dalam Polygon List.

```
    For A:=0 to ListCtr-1 do
```

Jika element Polygon List terpakai maka ...

```
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
```

```
        With Box[ObjNo] do
          Begin
```

Ambil titik pertama dari poligon yang sedang diakses sebagai titik start dari garis.

```
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
```

Lakukan loop untuk mengakses seluruh titik dalam polygon list yang sedang diakses.

```
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
```

Ambil titik berikut sebagai titik akhir dari garis.

```
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
```

Tampilkan garis di layar monitor dengan koordinat awal dan koordinat akhir yang telah ditentukan.

```
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
```

Pindahkan koordinat akhir dari garis yang terakhir digambar sebagai koordinat awal garis berikutnya. Dengan cara ini maka dihasilkan garis yang bersambung.

```
                  X1:=X2;              Y1:=Y2;
              End;
```

Lakukan penggambaran garis sekali lagi untuk menghubungkan titik terakhir dengan titik pertama dari poligon. Untuk itu ambil koordinat titik pertama dari poligon sebagai titik akhir (X2, Y2) dari garis karena secara otomatis, titik awal (X1,Y1) dari garis setelah keluar dari loop masih berisi informasi koordinat titik terakhir dari poligon.

```
              X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
              Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
              Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
            End;
      End;
  End;
```

Proyeksikan seluruh element Vertex Bank dalam objek dan masukkan poligon-poligon objek tersebut ke dalam Polygon List.

```
Procedure Project_Objek;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
```

```
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;
```

Selebihnya untuk Dilatasi, Rotasi dan Scaling tidak mengalami perubahan, mengingat yang kita transformasikan masih juga objeknya bukan poligon yang berada pada polygon list.

```
Procedure Main;
Var Ch:Char;
  Begin
    Repeat
      Repeat
```

Kosongkan Virtual Screen.

```
        FillWord(Buff^,32000,0);
```

Inisialisasikan Polygon List.

```
        Remove_PolyList;
```

Proyeksikan seluruh element Vertex Bank dari objek dan daftarkan poligon-poligonnya ke dalam Polygon List.

```
        Project_Objek;
```

Tampilkan poligon yang terdaftar pada Polygon List di layar monitor.

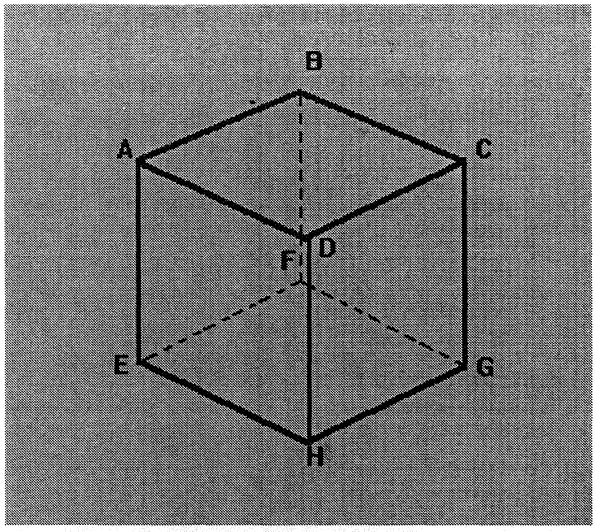
```
        Display_PList_WF;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
```

Lakukan proses transformasi sesuai dengan tombol keyboard yang ditekan.

```
      Case ch of
        'q':X_Rotation(2);
        'a':X_Rotation(-2);
        'w':Y_Rotation(2);
        's':Y_Rotation(-2);
        'e':Z_Rotation(2);
        'd':Z_Rotation(-2);
        'r':Scale(1.1,1,1);
        'f':Scale(0.9,1,1);
        't':Scale(1,1.1,1);
        'g':Scale(1,0.9,1);
        'y':Scale(1,1,1.1);
        'h':Scale(1,1,0.9);
        '7':Dilatasi(10,0,0);
        '1':Dilatasi(-10,0,0);
        '8':Dilatasi(0,10,0);
        '2':Dilatasi(0,-10,0);
        '9':Dilatasi(0,0,10);
        '3':Dilatasi(0,0,-10);
      End;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

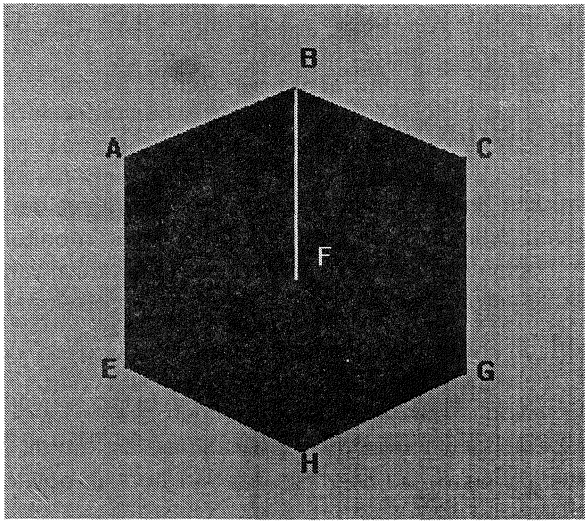
Ok, sekarang kita telah memiliki Polygon list yang berisi kumpulan poligon yang telah diproyeksikan. Polygon List untuk objek ABCD pada gambar sebelumnya berisi 4 poligon yaitu ABC, BDC, ACD dan ADB. Langkah selanjutnya ialah menampilkan keempat poligon ini pada layar monitor untuk melengkapi proyeksi tiga dimensi kita. Tetapi tunggu dulu, poligon mana yang harus kita tampilkan terlebih dahulu?

Kita tidak dapat menampilkan poligon-poligon tersebut semau kita karena hal tersebut akan mengakibatkan kesalahan proyeksi dari objek tersebut.

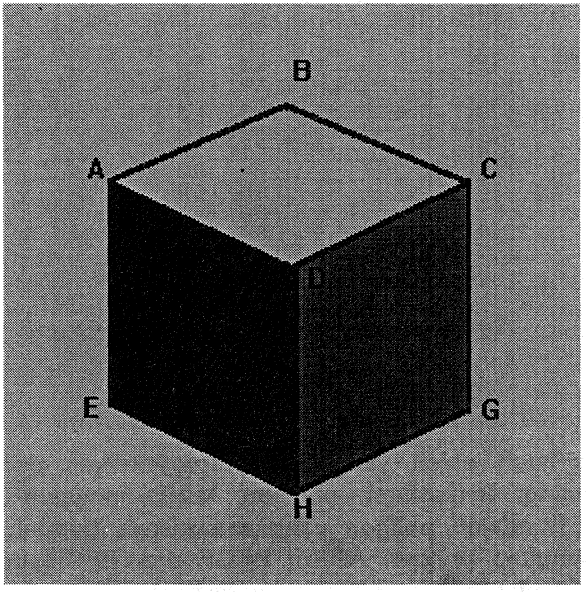


Ini adalah sebuah objek yang akan diproyeksikan dengan teknik polygon shading. Perhatikan, polygon AEHD, polygon CDHG dan polygon ADCB adalah poligon yang seharusnya tampak pada layar monitor sedangkan poligon ABFE, BCGF dan EFGH tidak tampak karena tertutup oleh ketiga poligon yang tampak.

Tetapi karena keenam poligon hasil proyeksi dari objek tersebut ditampilkan dengan tanpa urutan/semau kita, hasil dari proyeksi dapat berupa gambar seperti berikut.



Tentu saja ini hasil proyeksi yang salah, karena tidak seharusnya ketiga poligon tersebut yang tampak. Hasil proyeksi yang benar adalah seperti gambar berikut.



Bagaimana kita dapat mengetahui urutan dari penggambaran poligon, juga bagaimana cara mengetahui poligon yang mana yang tidak perlu ditampilkan? Hal-hal tersebut akan kita bahas secara mendetil pada bab selanjutnya.

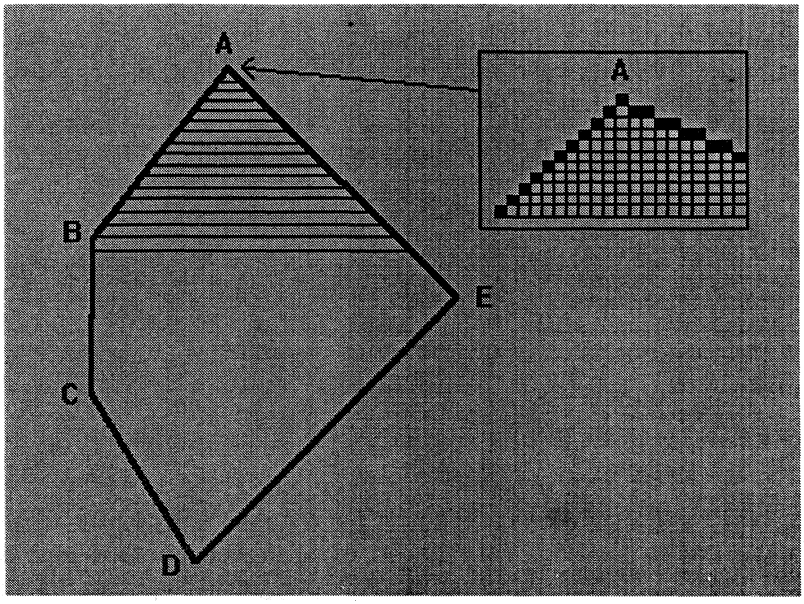
Untuk sementara kita asumsikan bahwa kita telah mengetahui urutan poligon yang harus kita tampilkan.

5.2 Polygon Shading

Pada bagian ini kita akan merakit sebuah algorithm yang bertujuan untuk mengarsir sebuah poligon hasil proyeksi pada layar monitor. Teknik ini sangat erat hubungannya dengan Bresenham algorithm yang telah kita pelajari pada buku pemrograman animasi dan game profesional seri kedua.

Sebuah poligon dibentuk oleh titik-titik yang dihubungkan dengan garis-garis. Dalam melakukan pengarsiran poligon, kita harus menerapkan algorithm yang paling efisien yang tidak membutuhkan banyak perhitungan sehingga mempercepat animasi yang akan dihasilkan. Untuk itu dilakukan sistem pengarsiran dari kiri ke kanan mulai dari puncak poligon hingga ke

bagian bawah poligon. Sehingga kita perlu menelusuri garis-garis dari poligon tersebut satu persatu.



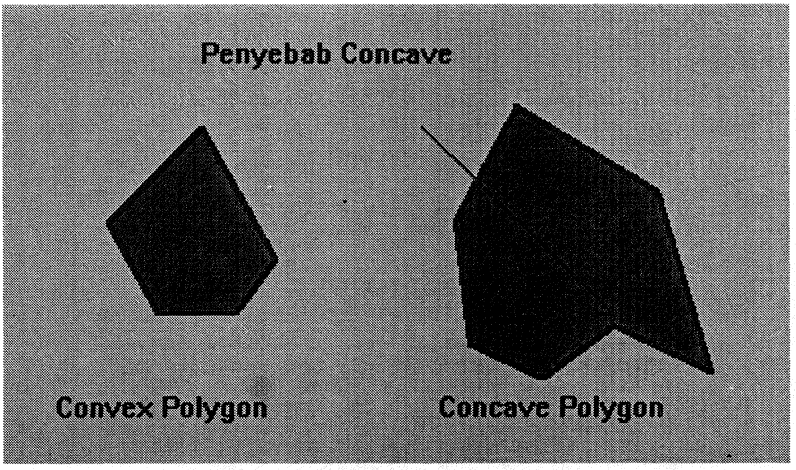
Gambar di atas ialah sebuah poligon yang diarsir mulai dari puncak. Window kecil pada bagian kanan menunjukkan gambar pembesaran dari apa yang sebenarnya terjadi pada layar monitor mengingat layar monitor adalah layar dot-matrix yang disusun dari balok-balok kecil.

Mula-mula dicari titik yang tertinggi. Di sini ialah titik A, kemudian dilakukan penelusuran kiri dan penelusuran kanan. Penelusuran kiri pada garis AB untuk mencari posisi horizontal awal untuk memulai pengarsiran. Sedangkan penelusuran kanan pada garis AE ditujukan untuk mencari posisi horizontal akhir dari pengarsiran. Setelah proses pengarsiran dilakukan, penelusuran kanan dan kiri dilakukan lagi untuk koordinat Y berikutnya (garis arsiran bawahnya).

Proses ini diulang terus hingga seluruh poligon telah diarsir, dengan kata lain koordinat Y yang sedang diproses sama dengan titik terendah (berharga Y paling besar) dari poligon tersebut.

Jika salah satu proses penelusuran telah sampai pada akhir garis yang ditelusuri, misalnya penelusuran kiri telah sampai pada titik B, maka penelusuran kiri dilakukan terhadap garis berikutnya yaitu garis BC. Demikian juga halnya bila penelusuran kanan mencapai titik E.

Teknik ini hanya berfungsi dengan benar jika poligon yang ditelusuri ialah convex polygon.



Pada Concave Polygon perlu dilakukan penanganan lebih lanjut sebelum dilakukan proses shading. Penanganan ini ialah dengan cara membagi poligon tersebut menjadi dua buah convex polygon. Tetapi cara ini tidak akan kita bahas pada edisi ini karena selain prosesnya cukup rumit, teknik ini membutuhkan CPU cycle yang cukup banyak sehingga dapat memperlambat jalannya animasi. Untuk mengatasi terjadinya Concave Polygon, jangan merancang objek tiga dimensi dengan menggunakan concave polygon. Karena concave polygon ada kemungkinan diproyeksikan menjadi concave polygon, sedangkan di lain pihak hasil proyeksi dari convex polygon pasti convex polygon juga.

Ok, waktu programming!!! Program di bawah ini adalah implementasi teknik polygon shading pada program B3PLIST.PAS.

```
{-----------------------------------------------                }
{ Program     : B3SHADE.PAS                                    }
{ Coder       : Agustinus Nalwan                               }
{ Description: Program transformasi objek                      }
{              dengan teknik Polygon Shading.                  }
{-----------------------------------------------                }

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
```

```
      Color       : Byte;
    End;

    Vertex_Rec = Record
      X, Y, Z : Real;
      Xp,Yp   : Longint;
    End;

    ObjRect = Record
      Color      : Byte;
      VertexBank : Array[0..20] of Vertex_Rec;
      Polygon    : Array[0..20] of Poly_Rec;
      TotalVertex: Integer;
      TotalPoly  : Integer;
      X,Y,Z      : Real;
    End;

    PolyList_Rec = Record
      ObjNo,PolyNo : Integer;
      Active       : Boolean;
    End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ListCtr:Integer;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=-lx;Box[No].VertexBank[0].Y:=ly;
       Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;Box[No].VertexBank[1].Y:=ly;
       Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
       Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
       Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;Box[No].VertexBank[4].Y:=ly;
       Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
       Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
       Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
       Box[No].VertexBank[7].Z:=lz;
    Box[No].Polygon[0].TotalVertex:=4;
```

```
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;

    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;

    Box[No].Polygon[0].Color:=17;
    Box[No].Polygon[1].Color:=19;
    Box[No].Polygon[2].Color:=21;
    Box[No].Polygon[3].Color:=23;
    Box[No].Polygon[4].Color:=25;
    Box[No].Polygon[5].Color:=27;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;
```

```
Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,150,0,850,20,50,30,11);
    Create_Box(2,-100,130,750,50,20,100,12);
    Remove_PolyList;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
            Inc(ListCtr);
          End;
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
```

```
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;         Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
    LeftCtr     : Integer;
    XL,XR,Bf    : Longint;
    Dx,Dy       : Real;
    Finish      : Boolean;
    Ty          : Longint;
  Begin
    YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMinIndex:=0;
    YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMaxIndex:=0;

    For A:=1 to Poly.TotalVertex-1 do
      Begin
        Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMinIndex:=A;
          End;
        If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMaxIndex:=A;
          End;
      End;

    LeftStart:=YMinIndex; RightStart:=YMinIndex;
    YCtr :=YMin;
    LeftEnd:=LeftStart+1;
    If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
    RightEnd:=RightStart-1;
    If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
```

```
    Finish:=False;
    While (Dy=0) and (Not Finish) do
      Begin
        LeftStart:=LeftStart+1;
        If LeftStart>=Poly.TotalVertex then LeftStart:=0;
        LeftEnd:=LeftStart+1;
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
        If LeftStart=YMaxIndex then Finish:=True;
      End;
    If Not Finish then
      Begin
        LeftSlope:=(Dx)/(Dy);
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
        While (Dy=0) and (Not Finish) do
          Begin
            RightStart:=RightStart-1;
            If RightStart<0 then RightStart:=Poly.TotalVertex-1;
            RightEnd:=RightStart-1;
            Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
            Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
            If RightStart=YMaxIndex then Finish:=True;
          End;
        RightSlope:=(Dx)/(Dy);
        XLeft :=Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].;
             Xp+LeftSlope;
        XRight:=Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].;
             Xp+RightSlope;
        While (Not Finish) do
          Begin
            XL:=Round(XLeft);
            XR:=Round(XRight);
            If XL>XR then
              Begin
                Bf:=XR;
                XR:=XL;
                XL:=Bf;
              End;

            If XL<0 then XL:=0;
            If XR>319 then XR:=319;
            If (YCtr>=0) and (Not Finish) then
             If XL=XR then PutPixel(Buff^,XL,YCtr,Poly.Color) else
              If XL<XR then
                HLines(Buff^,XL,XR,YCtr,Poly.Color);
            XLeft:=XLeft+LeftSlope;
            XRight:=XRight+RightSlope;
            Inc(YCtr);
            If YCtr>199 then Finish:=True;
            If Not Finish then
             If (YCtr>Box[ObjNo].VertexBank;
                [Poly.Vertex[LeftEnd]].Yp) then
              Begin
                Repeat
```

```
                    LeftStart:=LeftStart+1;
                    If LeftStart>=Poly.TotalVertex then LeftStart:=0;
                    LeftEnd:=LeftStart+1;
                    If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
                    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                      Xp-Box[ObjNo].VertexBank;
                         [Poly.Vertex[LeftStart]].Xp;
                    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                      Yp-Box[ObjNo].VertexBank;
                                 [Poly.Vertex[LeftStart]].Yp+1;
                    If LeftStart=YMaxIndex then Finish:=True;
                  Until (Dy<>0) or Finish;
                  If Not Finish then
                    Begin
                      LeftSlope:=(Dx)/(Dy);
                      XLeft:=Box[ObjNo].VertexBank;
                         [Poly.Vertex[LeftStart]].Xp+LeftSlope;
                    End;
                End;
              If Not Finish then
                Begin
                  If YCtr>Box[ObjNo].VertexBank
                     [Poly.Vertex[RightEnd]].Yp then
                    Begin
                      Repeat
                        RightStart:=RightStart-1;
                        If RightStart<0 then RightStart:=Poly.;
                         TotalVertex-1;
                        RightEnd:=RightStart-1;
                        If RightEnd<0 then RightEnd:=Poly.;
                         TotalVertex-1;
                        Dx:=Box[ObjNo].VertexBank[Poly.Vertex;
                         [RightEnd]].Xp-Box[ObjNo].VertexBank;
                           [Poly.Vertex[RightStart]].Xp;
                        Dy:=Box[ObjNo].VertexBank[Poly.Vertex;
                         [RightEnd]].Yp-Box[ObjNo].VertexBank;
                           [Poly.Vertex[RightStart]].Yp+1;
                        If RightStart=YMaxIndex then Finish:=True;
                      Until (Dy<>0) or Finish;
                      If Not Finish then
                        Begin
                          RightSlope:=(Dx)/(Dy);
                          XRight:=Box[ObjNo].VertexBank;
                             [Poly.Vertex[RightStart]].Xp;
                          XRight:=XRight+RightSlope;
                        End;
                    End;
                End;
            End;
        End;
    End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    ColCtr:=17;
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
```

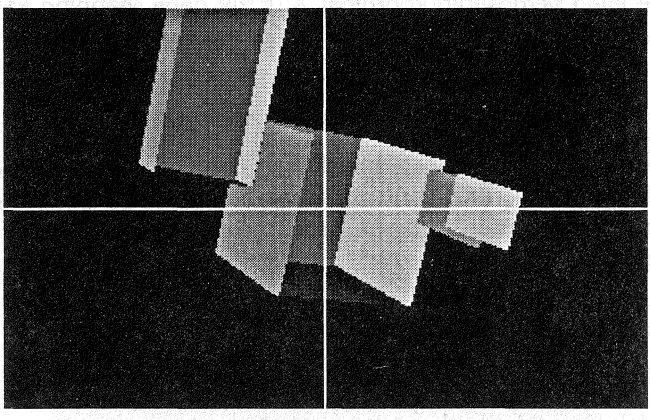
```
        Inc(ColCtr);
        If ColCtr>25 then ColCtr:=17;
        Poly_Shading(Box[ObjNo].Polygon[PolyNo],ObjNo);
      End;
  End;

Procedure Project_Objek;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
```

```
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        Box[A].X:=Box[A].X+Tx;
        Box[A].Y:=Box[A].Y+Ty;
        Box[A].Z:=Box[A].Z+Tz;
      End;
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Ch:=' ';
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Objek;
        BackFace_Filter;
        Display_PList_Shading;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          'r':Scale(1.1,1,1);
          'f':Scale(0.9,1,1);
          't':Scale(1,1.1,1);
          'g':Scale(1,0.9,1);
          'y':Scale(1,1,1.1);
          'h':Scale(1,1,0.9);
          '7':Dilatasi(10,0,0);
          '1':Dilatasi(-10,0,0);
          '8':Dilatasi(0,10,0);
          '2':Dilatasi(0,-10,0);
          '9':Dilatasi(0,0,10);
          '3':Dilatasi(0,0,-10);
        End;
      Until Keypressed;
      Ch:=Readkey;
```

```
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Coba jalankan program di atas. Oo ... Hal yang kita takuti ternyata terjadi. Coba rotasikan objek. Ketiga balok terlihat transparan. Efek ini sebenarnya terjadi karena poligon digambar dengan tidak berurutan. Juga poligon yang membelakangi kita ikut digambar sehingga merusak orientasi dari hasil proyeksi. Hmmm, tapi boleh juga ya buat efek transparan. (Memang kadang-kadang kesalahan program itu menyebalkan tetapi kadang juga bisa memberikan ilham-ilham yang baru.) Pembahasan mengenai masalah ini akan dilakukan pada bab berikutnya. Sekarang kita anggap saja program telah berjalan dengan benar (kan yang sedang kita rakit adalah poligon shading-nya, he-he-he).



Perhatikan. Objek terlihat transparan karena seluruh poligon digambar secara tidak berturutan. Juga ditampilkannya poligon yang tidak semestinya terlihat karena membelakangi kita. Tetapi poligon shading yang kita rakit telah berjalan dengan baik. Perhatikan sekarang seluruh poligon tidak berupa garis-garis tetapi telah diarsir.

PENJELASAN LISTING PROGRAM

Di sini kita gunakan sistem lokal koordinat untuk memproyeksikan objek sehingga hasil rotasi objek dapat terlihat lebih jelas, karena objek dirotasi terhadap titik pusatnya sendiri sehingga posisinya tidak berubah dan masih

terlihat di layar. Jika kita gunakan global koordinat, pada saat dirotasi, objek akan ikut bergeser sehingga sulit bagi kita untuk memonitornya.

Nah, marilah kita bahas bagian shading dari program di atas. Mengenai procedure-procedure lainnya tidak akan dijelaskan karena struktur pemrogramannya masih tetap sama persis seperti program kita sebelumnya.

Procedure inilah yang memegang peranan terpenting dalam melakukan proses shading. Adapun parameter yang dikirimkan ialah poligon yang hendak diarsir dan nomor objek yang memiliki poligon tersebut.

```
Procedure Poly_Shading(Poly:Poly_Rec;ObjNo:Integer);
Var A           : Integer;
```

LeftStart ialah variabel yang menyimpan nomor titik awal dari garis scanning sebelah kiri.

RightStart ialah variabel yang menyimpan nomor titik awal dari garis scanning sebelah kanan.

```
    LeftStart   : Integer;
    RightStart  : Integer;
```

RightSlope ialah pergeseran koordinat X untuk garis scanning kanan untuk setiap satu lompatan vertikal (gradient dari garis scanning kanan).

LeftSlope ialah pergeseran koordinat Y untuk garis scanning kiri untuk setiap satu lompatan vertikal (gradient dari garis scanning kiri).

Dengan cara ini, kita dapat melakukan penelusuran sepanjang garis scanning kanan dan kiri dengan cara menambah posisi sebelumnya dengan gradient-nya masing-masing tanpa perlu melakukan perhitungan perkalian dan pembagian.

```
    RightSlope  : Real;
    LeftSlope   : Real;
```

XLeft adalah koordinat horizontal awal dari posisi scanning.

Xright adalah koordinat horizontal akhir dari posisi scanning.

```
    XLeft,XRight: Real;
```

LeftEnd ialah variabel yang menyimpan nomor titik akhir dari garis scanning sebelah kiri.

RightEnd ialah variabel yang menyimpan nomor titik akhir dari garis scanning sebelah kanan.

```
    RightEnd    : Integer;
    LeftEnd     : Integer;
```

Yctr adalah variabel counter vertikal yang menunjukkan kordinat vertikal dari

arsiran yang sedang dilakukan.

```
YCtr          : Longint;
```

YMin dan YMax ialah koordinat Y terbesar dan terkecil yang dimiliki titik-titik yang membentuk poligon tersebut.

```
YMin,YMax     : Longint;
```

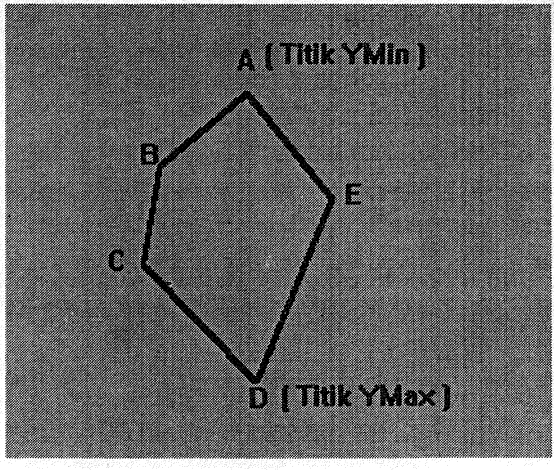
YMinIndex dan YMaxIndex ialah nomor titik yang koordinat Y-nya paling kecil dan paling besar.

```
YMinIndex     : Integer;
YMaxIndex     : Integer;
RightCtr      : Integer;
LeftCtr       : Integer;
XL,XR,Bf      : Longint;
Dx,Dy         : Real;
Finish        : Boolean;
Ty            : Longint;
Begin
```

Asumsikan, koordinat terbesar dan terkecil dipegang oleh titik pertama dari poligon.

```
YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
YMinIndex:=0;
YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
YMaxIndex:=0;
```

Lakukan scanning terhadap seluruh titik yang menghubungkan poligon untuk mencari titik yang memiliki koordinat Y terkecil sebagai start awal dari proses pengarsiran, dan titik yang memiliki koordinat Y terbesar sebagai akhir dari proses pengarsiran.



```
For A:=1 to Poly.TotalVertex-1 do
  Begin
    Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
```

Update YMin jika ditemukan titik yang berkoordinat Y lebih kecil dari YMin sekarang.

```
    If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
      Begin
        YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        YMinIndex:=A;
      End;
```

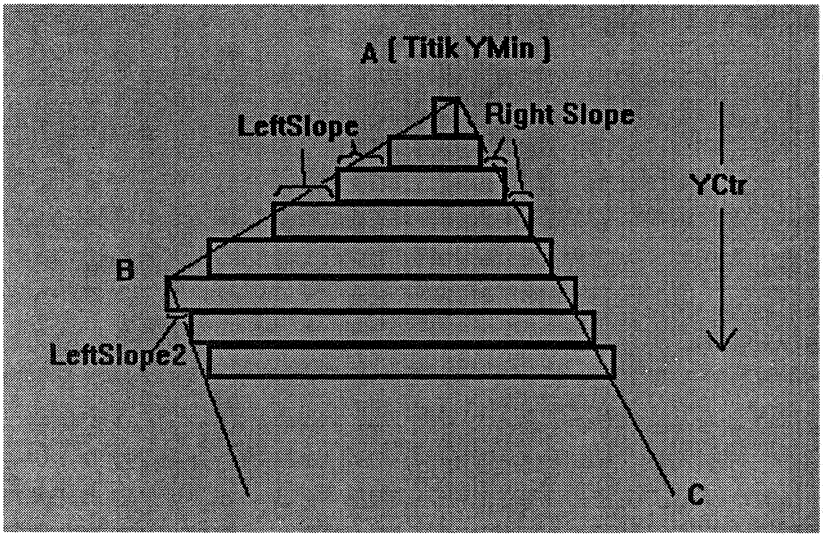
Update YMax jika ditemukan titik yang memiliki koordinat Y lebih besar dari YMax sekarang.

```
    If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
      Begin
        YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        YMaxIndex:=A;
      End;
  End;
```

Titik awal garis scanning kiri dimulai dari titik teratas (titik pemegang YMin), demikian pula dengan titik awal garis scanning kanan.

```
LeftStart:=YMinIndex; RightStart:=YMinIndex;
YCtr :=YMin;
```

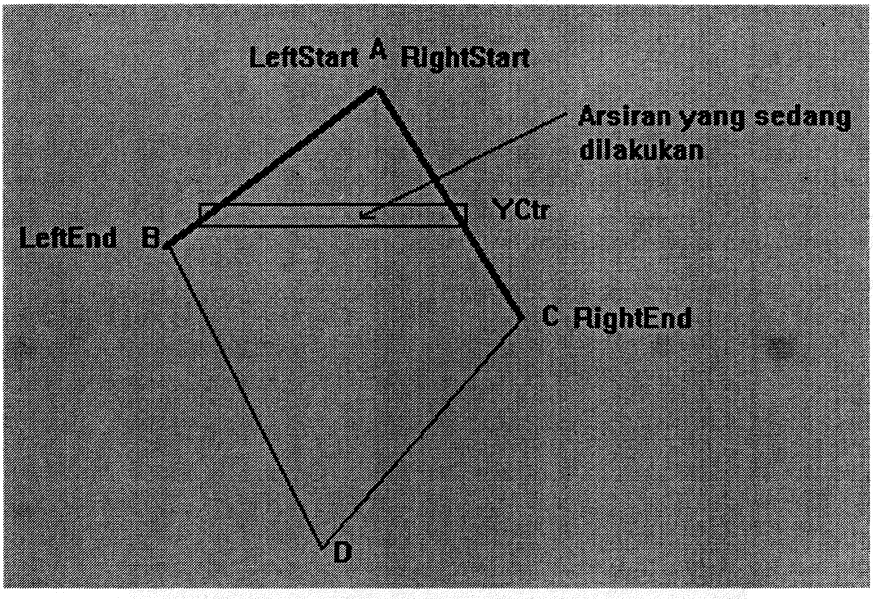
YCtr yang menunjukkan koordinat Y dari arsiran yang sekarang sedang dilakukan diinisialisasikan menjadi YMin (koordinat Y teratas).



Perhatikan gambar di atas. Masing-masing persegi panjang yang disusun vertikal itu ialah arsiran perbaris horizontal yang dilakukan. YCtr menunjukkan koordinat Y dari arsiran yang sedang dilakukan. YCtr ini dimulai dari titik YMin. LeftSlope ialah pergeseran koordinat X pada garis scanning kiri yang sedang dipakai (AB) jika koordinat Y bertambah 1. LeftSlope2 ialah pergeseran koordinat X pada garis scanning kiri berikutnya. (BD).

RightSlope ialah pergeseran koordinat X pada garis scanning kanan yang sedang dipakai (AC) jika koordinat Y bertambah 1.

Pada posisi awal, LeftStart akan menunjuk pada titik A. Dan LeftEnd akan menunjuk pada LeftStart + 1 yaitu titik B (karena susunan titik dilakukan berlawanan arah jarum jam). Sedangkan RightStart akan menunjuk pada titik A juga, dan RightEnd akan menunjuk pada titik C.



Gambar ini menunjukkan, posisi awal dari proses pengarsiran setelah dilakukan beberapa baris pengarsiran.

```
LeftEnd:=LeftStart+1;
If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
RightEnd:=RightStart-1;
If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
```

Ambil panjang horizontal dari garis scanning kiri (garis LeftStart ke RightStart atau pada situasi awal ialah garis AB).

```
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
```

Ambil panjang vertikal dari garis scanning kiri.

```
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
    Finish:=False;
```

Jika selisih vertikal = 0 maka lakukan scanning pada garis kiri berikutnya (BD). Karena jika selisih vertikal = 0 berarti YCtr tidak akan bertambah, jadi tidak perlu dilakukan scanning.

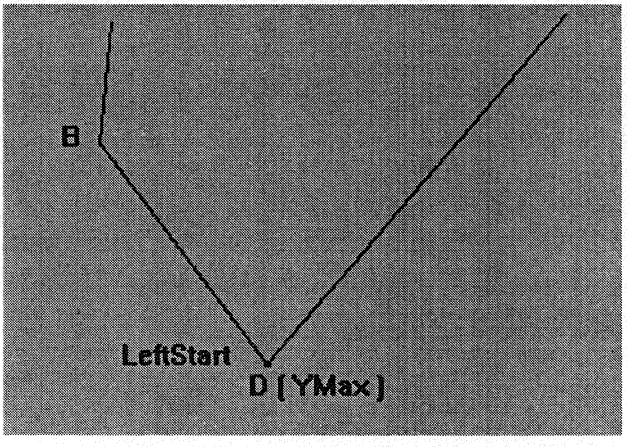
```
  While (Dy=0) and (Not Finish) do
    Begin
      LeftStart:=LeftStart+1;
      If LeftStart>=Poly.TotalVertex then LeftStart:=0;
      LeftEnd:=LeftStart+1;
```

Lakukan perhitungan panjang vertikal dan horizontal dari garis karena garis scanning yang diproses telah berubah.

```
      Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
        Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
      Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
        Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
```

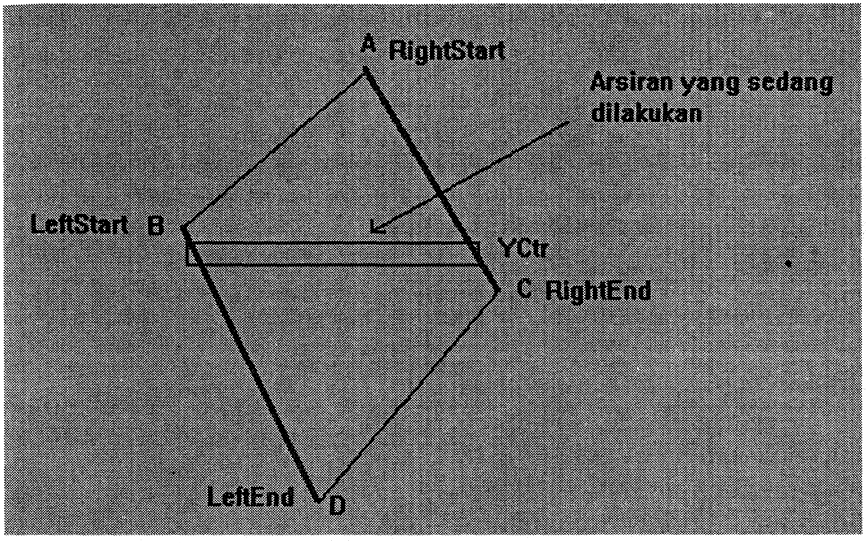
Jika awal garis scanning kiri telah mencapai titik yang memiliki koordinat terendah, maka proses selesai.

Situasi ini digambarkan pada diagram di bawah ini.




```
      If LeftStart=YMaxIndex then Finish:=True;
    End;
  If Not Finish then
    Begin
```

Lakukan perhitungan gradient yang baru karena scanning kiri telah berpindah ke garis berikutnya (BD).



Perhatikan, gradient scanning kiri diperbarui karena sekarang garis scanning telah berpindah dari AB ke BD.

```
LeftSlope:=(Dx)/(Dy);
```

Lakukan perhitungan panjang vertikal dan horizontal garis scanning kanan.

```
Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
  Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
  Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
```

Jika selisih vertikal = 0 maka pindah ke garis scanning kanan berikutnya.

```
While (Dy=0) and (Not Finish) do
  Begin
    RightStart:=RightStart-1;
    If RightStart<0 then RightStart:=Poly.TotalVertex-1;
    RightEnd:=RightStart-1;
```

Update perhitungan panjang vertikal dan horizontal garis scanning kanan karena garis scanning kanan telah berpindah.

```
Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
 Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
 Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
```

Proses selesai jika Start awal dari garis scanning kanan telah mencapai YMax.

```
    If RightStart=YMaxIndex then Finish:=True;
  End;
```

Lakukan perhitungan gradient untuk garis scanning kanan yang baru.

```
RightSlope:=(Dx)/(Dy);
```

Set koordinat X awal dan akhir dari garis arsiran. XLeft dan XRight ini akan selalu bergeser mengikuti garis scanning kiri dan garis scanning kanan dalam rangka mengarsir seluruh poligon.

```
Xleft:=Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].;
  Xp+LeftSlope;
XRight:=Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].;
  Xp+RightSlope;
```

Lakukan loop untuk mengarsir poligon dari YMin hingga YMax.

```
While (Not Finish) do
  Begin
    XL:=Round(XLeft);
    XR:=Round(XRight);
```

Jika terjadi XL lebih besar dari XR maka tukar XL dan XR. Sebenarnya kejadian ini terjadi jika poligon membelakangi kita sehingga aturan titik dari poligon akan menjadi searah jarum jam (ingat pada saat kita membahas poylgon shading data structure). Sehingga XLeft akan terbalik dengan XRight (XLeft adalah garis scanning kanan dan XRight adalah garis scanning kiri).

```
    If XL>XR then
      Begin
        Bf:=XR;
        XR:=XL;
        XL:=Bf;
      End;
```

Jika XL melebihi batas kiri layar, maka set XL = 0.

```
    If XL<0 then XL:=0;
```

Jika XR melebihi batas kanan layar, maka set XR = 319.

```
    If XR>319 then XR:=319;
```

Jika YCtr masih berada pada jangkauan layar (lebih besar atau sama dengan 0) maka ...

```
    If (YCtr>=0) and (Not Finish) then
```

Jika terjadi XL = XR, garis yang hendak diarsir berupa sebuah titik maka lakukan PutPixel untuk mempercepat kalkulasi, mengingat perhitungan awal

untuk fungsi penggambaran garis lebih panjang dari penggambaran titik.

```
If XL=XR then PutPixel(Buff^,XL,YCtr,Poly.Color) else
```

Pastikan XL < XR.

```
If XL<XR then
```

Arsir garis dari XL ke XR dengan koordinat Y = YCtr.

```
HLines(Buff^,XL,XR,YCtr,Poly.Color);
```

Pindah ke koordinat Y berikutnya dengan menggeser XLeft dan XRight sepanjang garis scanning kiri dan kanan sekarang. Penggeseran cukup dilakukan dengan menambahkan posisi sekarang dengan gradient-nya masing-masing. Jangan lupa menambah YCtr.

```
XLeft:=XLeft+LeftSlope;
XRight:=XRight+RightSlope;
Inc(YCtr);
```

Jika YCtr telah melebihi batas bawah layar, maka proses selesai.

```
If YCtr>199 then Finish:=True;
If Not Finish then
```

Jika YCtr telah mencapai LeftEnd maka pindahkan garis scanning kiri ke garis selanjutnya (BD).

```
If(YCtr>Box[ObjNo].VertexBank;
  [Poly.Vertex[LeftEnd]].Yp) then
 Begin
   Repeat
     LeftStart:=LeftStart+1;
     If LeftStart>=Poly.TotalVertex then LeftStart:=0;
     LeftEnd:=LeftStart+1;
     If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
```

Lakukan perhitungan panjang vertikal dan horizontal dari garis scanning kiri yang baru.

```
     Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
        Xp-Box[ObjNo].VertexBank;
           [Poly.Vertex[LeftStart]].Xp;
     Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
        Yp-Box[ObjNo].VertexBank;
           [Poly.Vertex[LeftStart]].Yp+1;
     If LeftStart=YMaxIndex then Finish:=True;
   Until (Dy<>0) or Finish;
   If Not Finish then
     Begin
```

Lakukan perhitungan gradient dari garis scanning kiri yang baru.

```
LeftSlope:=(Dx)/(Dy);
```

Set XLeft menjadi titik awal dari garis scanning kiri.

```
XLeft:=Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp+LeftSlope;
                End;
            End;
          If Not Finish then
            Begin
```

Lakukan proses yang sama terhadap garis scanning kanan.

```
              If YCtr>Box[ObjNo].VertexBank;
                 [Poly.Vertex[RightEnd]].Yp then
                Begin
                  Repeat
                    RightStart:=RightStart-1;
                    If RightStart<0 then;
                      RightStart:=Poly. TotalVertex-1;
                    RightEnd:=RightStart-1;
                    If RightEnd<0 then;
                      RightEnd:=Poly.TotalVertex-1;
```

Lakukan kalkulasi panjang vertikal dan horizontal dari garis scanning kanan yang baru.

```
            Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
                   Xp-Box[ObjNo].VertexBank;
                     [Poly.Vertex[RightStart]].Xp;

            Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
                   Yp-Box[ObjNo].VertexBank;
                     [Poly.Vertex[RightStart]].Yp+1;
                   If RightStart=YMaxIndex then Finish:=True;
                 Until (Dy<>0) or Finish;
                 If Not Finish then
                   Begin
```

Lakukan perhitungan gradient yang baru untuk garis scanning kanan.

```
                     RightSlope:=(Dx)/(Dy);
```

Set garis XRight menjadi awal dari garis scanning kanan.

```
                      XRight:=Box[ObjNo].VertexBank;
                        [Poly.Vertex[RightStart]].Xp;
                      XRight:=XRight+RightSlope;
                    End;
                End;
            End;
        End;
    End;
End;
```

Modifikasi berikutnya ialah penambahan procedure di bawah ini. Tujuannya

ialah untuk mengirimkan element-element dari polygon list satu persatu ke Poly_Shading untuk diarsir.

```
Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
```

Lakukan scanning terhadap seluruh poligon yang terdaftar dalam poligon list.

```
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
```

Panggil Poly_Shading untuk mengarsir poligon yang sedang diakses.

```
        Poly_Shading(Box[ObjNo].Polygon[PolyNo],ObjNo);
      End;
  End;
```

Teknik Polygon Shading ini sangat penting untuk dipahami karena merupakan dasar dari teknik-teknik proyeksi yang lainnya, seperti Flat Shading, Goraud Shading, Pong Shading dan Texture Mapping yang sedang gencar-gencarnya dipergunakan di dunia game tiga dimensi saat ini. Apalagi dengan munculnya processor pentium dengan chip clock yang sangat cepat, sehingga kalkulasi yang dulunya menjadi penghambat optimasi kecepatan, sekarang bukanlah masalah. Hal ini terbukti dengan mulai bermunculannya game-game tiga dimensi seperti DUKE NUKEM 3D, TERMINAL VELOCITY, game-game racing seperti BIG RED RACING, DESTRUCTION DERBY, THE NEED FOR SPEED dan game ID software (pembuat DOOM) yang baru yang bakal mengguncangkan teknik pemrograman game yaitu QUAKE. Sekilas tentang QUAKE, game ini menggunakan teknik polygon shading yang didukung oleh pencahayaan yang benar-benar mengagumkan yang bernama Dynamic Light Sourcing.

Teknologi game semakin lama semakin berkembang. Untuk itu janganlah mau ketinggalan!!! Kejar teknologi-teknologi tersebut dan bahkan kalau bisa ... KALAHKAN!!!

BAB 6

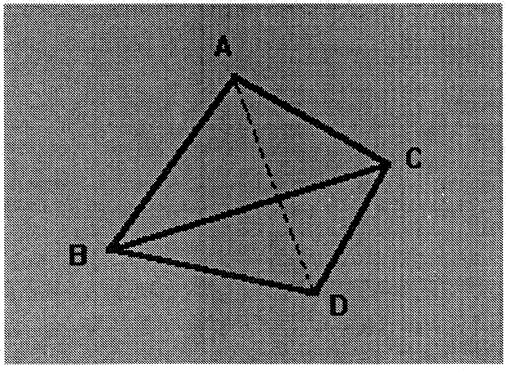
3D FILTERING

Seperti yang telah kita bahas pada bab sebelumnya, pekerjaan kita tidak selesai setelah polygon list terbentuk. Langkah selanjutnya ialah melakukan proses filtering. Proses ini dilakukan untuk mengurangi jumlah poligon yang perlu ditampilkan pada layar monitor, juga untuk mengurutkan poligon yang hendak ditampilkan sesuai dengan urutan yang benar.

Dalam pemrograman tiga dimensi, dikenal beberapa proses filter yang harus dilakukan.

6.1 Backface Filtering

Backface filtering ialah filter untuk menghilangkan poligon yang membelakangi kita.



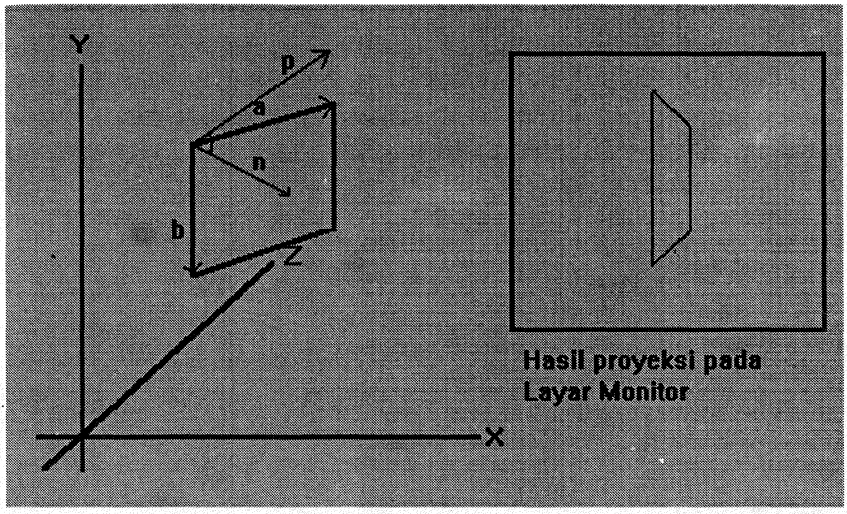
Perhatikan gambar objek di atas. Poligon ACD dan poligon ADB adalah poligon yang membelakangi kita, sedangkan poligon ABC dan BDC adalah poligon yang menghadap ke arah kita. Poligon yang membelakangi kita tidak perlu kita tempilkan karena secara otomatis, poligon tersebut pastilah tertutup oleh poligon yang menghadap ke arah kita. Oleh karena itu kita dapat menghapus semua poligon yang membelakangi kita di dalam poligon list sehingga mengurangi jumlah poligon yang perlu ditampilkan.

6.2 Cross Product Filtering

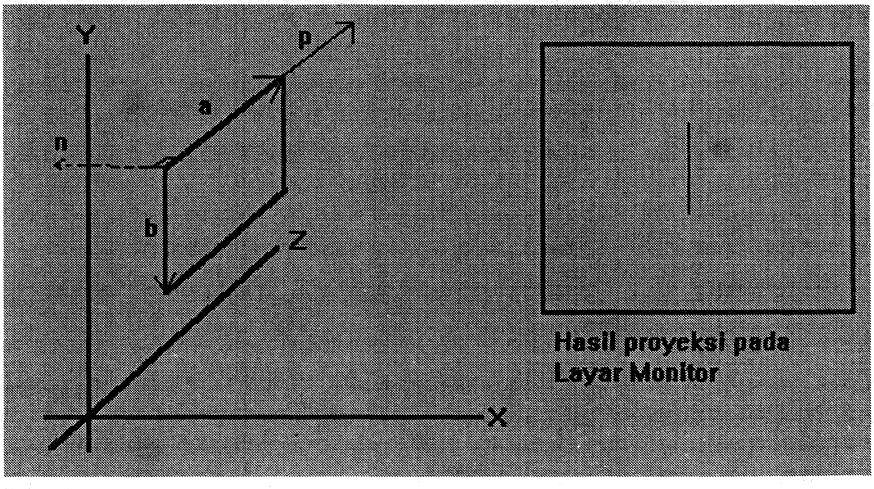
Cara mengetahui apakah suatu poligon membelakangi kita atau tidak, ada beberapa macam. Cara yang pertama, yang paling mudah dan membutuhkan cukup banyak proses perhitungan ialah dengan menggunakan cross product.

Mula-mula kita ambil dua buah garis yang berhenti pada satu titk dari poligon yang sedang kita periksa sebagai dua buah vektor. Kemudian lakukan cross product antara keduanya. Dengan cara ini ditemukan normal vektor dari poligon yang sedang kita periksa. Normal Vektor/Garis Normal ialah satu-satunya garis yang tegak lurus terhadap poligon. Garis ini menunjukkan arah ke mana poligon tersebut menghadap. Langkah selanjutnya ialah menghitung sudut antara garis normal tersebut dengan garis sudut pandang kita terhadap titik di mana diambil dua buah vektor tadi (garis ini disebut sebagai garis pandang). Dalam proses perhitungan sudut antara kedua vektor, dilakukan dot product antara kedua vektor dan disusul dengan pembagian dengan magnitude dari kedua vektor. Untuk mengetahui apakah sebuah poligon menghadap atau membelakangi kita, kita tidak perlu melakukan perhitungan hingga sampai ketemu sudutnya, tetapi cukup dengan melakukan Dot Product antara kedua vektor tanpa membaginya dengan magnitude. Jika hasil Dot Product dari garis normal dan garis pandang lebih kecil dari 0 maka sudut antara keduanya pasti sudut obtuse (lebih besar dari 90 derajat) maka poligon pasti menghadap kita. Jika hasil dot product antara garis normal dan garis pandang lebih besar atau sama dengan 0 maka sudut antara keduanya pasti sudut acute atau lebih kecil dari 90 derajat, maka poligon dianggap membelakangi kita. Teknik ini telah kita pelajari di Bab 3 pada bagian Vektor.

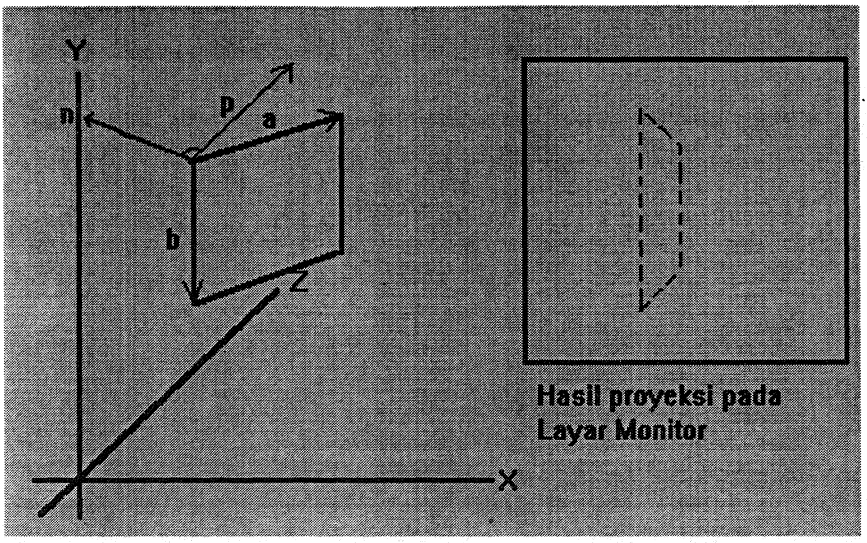
Sekarang marilah kita menganalisa beberapa kasus dengan algorithm yang baru kita pelajari.



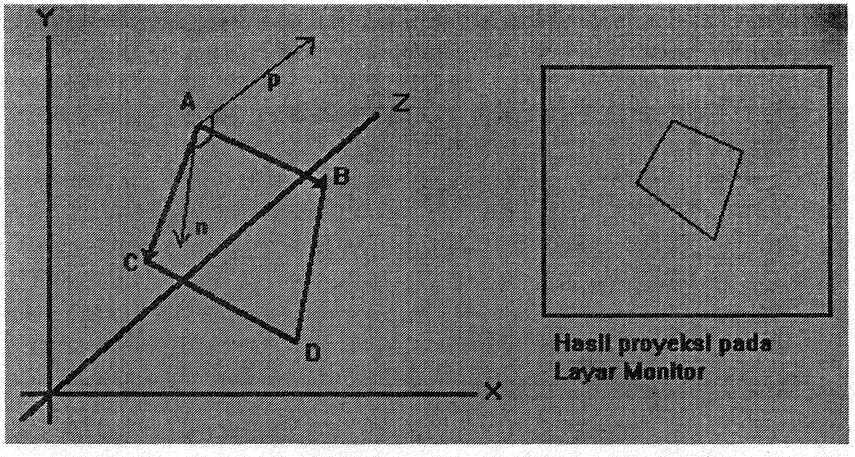
Garis P adalah garis sudut pandang kita yang ditarik dari mata kita ke titik di mana kita mendapatkan dua buah vektor a dan b pada poligon yang sedang kita periksa. Garis n adalah garis normal dari poligon yang hendak diproyeksikan. Di sini sudut antara p dan n setelah dihitung secara vektor, berharga sekitar 100 derajat (kira-kira). Dengan perhitungan dot product, didapatkan hasil negatif, maka poligon menghadap ke arah kita dan poligon ditampilkan di layar.



Di sini sudut antara garis penglihatan kita (p) dengan normal vektor (n) membentuk sudut 90 derajat. Hasil perhitungan dot product berharga 0. Berarti poligon ini masih dianggap menghadap ke arah kita.



Pada situasi ini, sudut antara p dan n lebih kecil dari 90 derajat. Katakanlah 80 derajat. Hasil perhitungan dot product menunjukkan angka positif, maka poligon dalam keadaan membelakangi kita. Untuk itu poligon ini tidak akan ditampilkan pada layar monitor.



Pada situasi ini p dan n membentuk sudut sekitar 160 derajat. Hasil perhitungan dot product angka negatif, maka poligon berada pada keadaan menghadap ke arah kita. Untuk itu poligon ini tetap ditampilkan pada layar monitor.

Nah, telah terbukti bahwa algorithm BACKFACE filtering kita yang pertama, berfungsi dengan benar. Mari kita coba teknik ini dengan menerapkannya pada program.

```
{-----------------------------------------------}
{ Program     : B3SHADE3.PAS                    }
{ Coder       : Agustinus Nalwan                }
{ Description: Program transformasi object      }
{              dengan filter BackFace normal    }
{              calculation                      }
{-----------------------------------------------}

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
       Xp,Yp   : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
       X,Y,Z      : Real;
     End;

     PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
       ZMin         : Real;
       ZMax         : Real;
       ZAvg         : Real;
     End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ListCtr:Integer;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;
```

```
Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=-lx;Box[No].VertexBank[0].Y:=ly;
        Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;Box[No].VertexBank[1].Y:=ly;
        Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
        Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
        Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;Box[No].VertexBank[4].Y:=ly;
        Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
        Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
        Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
        Box[No].VertexBank[7].Z:=lz;
    Box[No].Polygon[0].TotalVertex:=4;
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;

    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;

    Box[No].Polygon[0].Color:=17;
    Box[No].Polygon[1].Color:=19;
    Box[No].Polygon[2].Color:=21;
```

```
    Box[No].Polygon[3].Color:=23;
    Box[No].Polygon[4].Color:=25;
    Box[No].Polygon[5].Color:=27;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,90,0,850,20,50,30,11);
    Create_Box(2,80,130,850,50,20,100,12);
    Remove_PolyList;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
            PolygonList[ListCtr].ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
```

```
            PolygonList[ListCtr].ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr].ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr].ZMin>VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;
              End;
            PolygonList[ListCtr].ZAvg:=(PolygonList[ListCtr].;
               ZMax+PolygonList[ListCtr].ZMin)/2;
            Inc(ListCtr);
          End;
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;         Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
```

```
  LeftCtr      : Integer;
  XL,XR,Bf     : Longint;
  Dx,Dy        : Real;
  Finish       : Boolean;
  Ty           : Longint;
Begin
  YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
  YMinIndex:=0;
  YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
  YMaxIndex:=0;

  For A:=1 to Poly.TotalVertex-1 do
    Begin
      Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
      If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
        Begin
          YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
          YMinIndex:=A;
        End;
      If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
        Begin
          YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
          YMaxIndex:=A;
        End;
    End;

  LeftStart:=YMinIndex; RightStart:=YMinIndex;
  YCtr :=YMin;
  LeftEnd:=LeftStart+1;
  If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
  RightEnd:=RightStart-1;
  If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
  Dx:=Box[ObjNo].VertexBank;[Poly.Vertex[LeftEnd]].;
     Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
  Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
     Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
  Finish:=False;
  While (Dy=0) and (Not Finish) do
    Begin
      LeftStart:=LeftStart+1;
      If LeftStart>=Poly.TotalVertex then LeftStart:=0;
      LeftEnd:=LeftStart+1;
      Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
          Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
      Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
          Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
      If LeftStart=YMaxIndex then Finish:=True;
    End;
  If Not Finish then
    Begin
      If Dy<0 then
        Dy:=Dy;
      LeftSlope:=(Dx)/(Dy);
      Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
        Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
      Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
        Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
      While (Dy=0) and (Not Finish) do
        Begin
          RightStart:=RightStart-1;
```

```
    If RightStart<0 then RightStart:=Poly.TotalVertex-1;
    RightEnd:=RightStart-1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
       Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
       Yp-Box[ObjNo].VertexBank;
        [Poly.Vertex[RightStart]].Yp+1;
    If RightStart=YMaxIndex then Finish:=True;
  End;
If Dy<0 then
  Dy:=Dy;
RightSlope:=(Dx)/(Dy);
XLeft :=Box[ObjNo].VertexBank;
   [Poly.Vertex[LeftStart]].Xp+LeftSlope;
XRight:=Box[ObjNo].VertexBank[Poly.Vertex;
   [RightStart]].Xp+RightSlope;
While (Not Finish) do
  Begin
    XL:=Round(XLeft);
    XR:=Round(XRight);

    If XL>XR then
      Begin
        Bf:=XR;
        XR:=XL;
        XL:=Bf;
      End;

    If XL<0 then XL:=0;
    If XR>319 then XR:=319;
    If (YCtr>=0) and (Not Finish) then
     If XL=XR then PutPixel(Buff^,XL,YCtr,Poly.Color) else
      If XL<XR then
        HLines(Buff^,XL,XR,YCtr,Poly.Color);
    XLeft:=XLeft+LeftSlope;
    XRight:=XRight+RightSlope;
    Inc(YCtr);
    If YCtr>199 then Finish:=True;
    If Not Finish then
     If (YCtr>Box[ObjNo].VertexBank;
       [Poly.Vertex[LeftEnd]].Yp) then
      Begin
        Repeat
          LeftStart:=LeftStart+1;
          If LeftStart>=Poly.TotalVertex then LeftStart:=0;
          LeftEnd:=LeftStart+1;
          If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
          Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
               Xp-Box[ObjNo].VertexBank;
                [Poly.Vertex[LeftStart]].Xp;
          Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
               Yp-Box[ObjNo].VertexBank;
                [Poly.Vertex[LeftStart]].Yp+1;
          If LeftStart=YMaxIndex then Finish:=True;
        Until (Dy<>0) or Finish;
        If Not Finish then
          Begin
            If Dy<0 then
              Dy:=Dy;
            LeftSlope:=(Dx)/(Dy);
```

```
                    XLeft:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[LeftStart]].Xp+LeftSlope;
                  End;
              End;
            If Not Finish then
              Begin
                If YCtr>Box[ObjNo].VertexBank;
                  [Poly.Vertex[RightEnd]].Yp then
                  Begin
                    Repeat
                      RightStart:=RightStart-1;
                      If RightStart<0 then;
                        RightStart:=Poly.TotalVertex-1;
                      RightEnd:=RightStart-1;
                      If RightEnd<0 then;
                        RightEnd:=Poly.TotalVertex-1;
                      Dx:=Box[ObjNo].VertexBank;
                        [Poly.Vertex[RightEnd]].Xp-Box[ObjNo].;
                          VertexBank[Poly.Vertex[RightStart]].Xp;
                      Dy:=Box[ObjNo].VertexBank;
                        [Poly.Vertex[RightEnd]].Yp-Box[ObjNo].;
                          VertexBank[Poly.Vertex[RightStart]].Yp+1;
                      If RightStart=YMaxIndex then Finish:=True;
                    Until (Dy<>0) or Finish;
                    If Not Finish then
                      Begin
                        If Dy<0 then
                          Dy:=Dy;
                        RightSlope:=(Dx)/(Dy);
                        XRight:=Box[ObjNo].VertexBank;
                          [Poly.Vertex[RightStart]].Xp;
                        XRight:=XRight+RightSlope;
                      End;
                  End;
              End;
          End;
      End;
  End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        Poly_Shading(Box[ObjNo].Polygon[PolyNo],ObjNo);
      End;
  End;

Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
  Begin
    For Ctr:=0 to ListCtr-1 do
      Begin
        ObjNo :=PolygonList[Ctr].ObjNo;
        PolyNo:=PolygonList[Ctr].PolyNo;
```

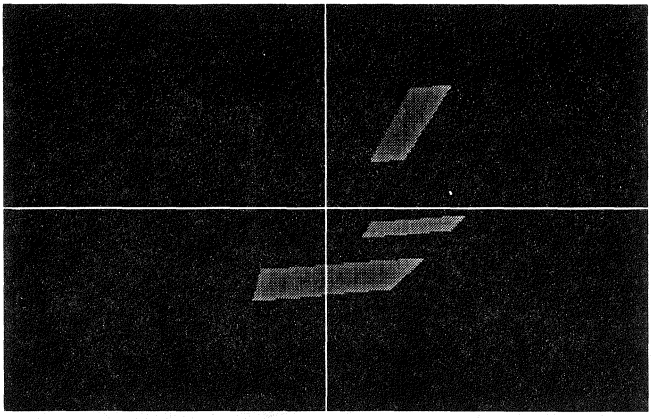
```
        A[0]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[0]].X-
               Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[1]].X;
        A[1]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[0]].Y-
               Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[1]].Y;
        A[2]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[0]].Z-
               Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[1]].Z;

        View[0]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[1]].X;
        View[1]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[1]].Y;
        View[2]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[1]].Z+400;

        B[0]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[2]].X-
               Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[1]].X;
        B[1]:=Box[ObjNo].VertexBank[Box[ObjNo].;
           Polygon[PolyNo].Vertex[2]].Y-
               Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[1]].Y;
        B[2]:=Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[2]].Z-
               Box[ObjNo].VertexBank[Box[ObjNo].;
                 Polygon[PolyNo].Vertex[1]].Z;
        Vect_Cross(Normal,A,B);
        If Vect_Dot(Normal,View)>=0 then;
          PolygonList[Ctr].Active:=False;
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
```

```
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        Box[A].X:=Box[A].X+Tx;
        Box[A].Y:=Box[A].Y+Ty;
        Box[A].Z:=Box[A].Z+Tz;
      End;
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Ch:=' ';
```

```
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Object;
        BackFace_Filter;
        Display_PList_Shading;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          'r':Scale(1.1,1,1);
          'f':Scale(0.9,1,1);
          't':Scale(1,1.1,1);
          'g':Scale(1,0.9,1);
          'y':Scale(1,1,1.1);
          'h':Scale(1,1,0.9);
          '7':Dilatasi(10,0,0);
          '1':Dilatasi(-10,0,0);
          '8':Dilatasi(0,10,0);
          '2':Dilatasi(0,-10,0);
          '9':Dilatasi(0,0,10);
          '3':Dilatasi(0,0,-10);
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Rotasikan ketiga kubus tersebut, perhatikan bahwa efek transparan sudah tidak terlihat lagi. Poligon-poligon yang membelakangi kita tidak nampak lagi.



Poligon yang posisinya membelakangi kita tidak lagi ditampilkan.

PENJELASAN LISTING PROGRAM

Sekarang marilah kita membahas implementasi dari algorithm BackFace Filtering pada program kita.

Tentunya Anda masih ingat dengan penggunaan Polygon List pada program-program kita terdahulu. Sebenarnya setelah sebuah poligon dianalisa, jika poligon tersebut tidak perlu ditampilkan maka poligon tersebut cukup dicoret dari Polygon List.

Procedure di bawah inilah yang melakukan proses BackFace Filtering.

```
Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
  Begin
```

Lakukan scanning terhadap seluruh poligon dalam polygon list.

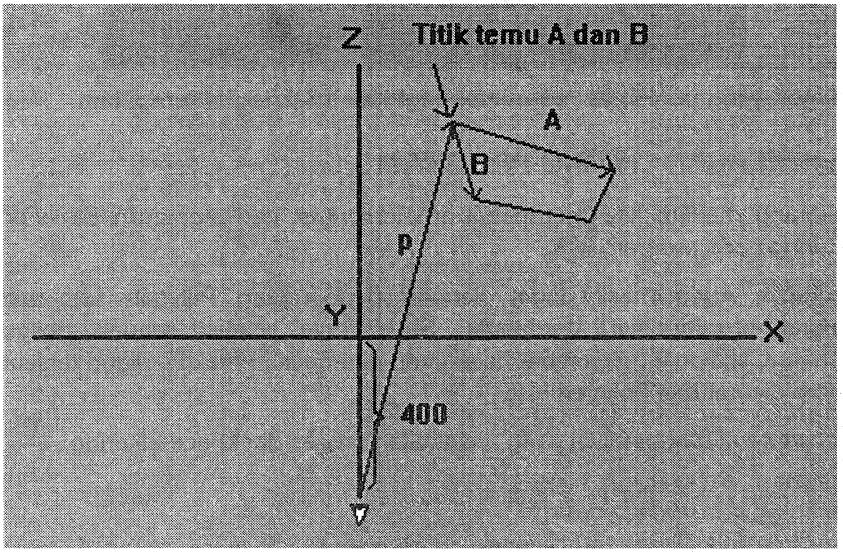
```
    For Ctr:=0 to ListCtr-1 do
      Begin
        ObjNo :=PolygonList[Ctr].ObjNo;
        PolyNo:=PolygonList[Ctr].PolyNo;
```

Ambil titik kedua dan titik pertama sebagai vektor pertama (Vektor A) dari poligon.

```
        A[0]:=Box[ObjNo].VertexBank[Box[ObjNo].;
          Polygon[PolyNo].Vertex[0]].X-
              Box[ObjNo].VertexBank[Box[ObjNo].;
                Polygon[PolyNo].Vertex[1]].X;
        A[1]:=Box[ObjNo].VertexBank[Box[ObjNo].;
          Polygon[PolyNo].Vertex[0]].Y-
              Box[ObjNo].VertexBank[Box[ObjNo].;
                Polygon[PolyNo].Vertex[1]].Y;
```

```
A[2]:=Box[ObjNo].VertexBank[Box[ObjNo].;
  Polygon[PolyNo].Vertex[0]].Z-
      Box[ObjNo].VertexBank[Box[ObjNo].;
        Polygon[PolyNo].Vertex[1]].Z;
```

Karena garis pandang yang hendak kita cari ialah garis yang menghubungkan mata kita dengan titik di mana dua buah vektor bertemu (dalam hal ini vektor A dan vektor B), maka vektor garis pandang itu tidak lain ialah koordinat dari titik kedua (di mana A dan B bertemu) yang koordinat Z-nya ditambah dengan 400 karena jarak antara origin (0,0,0) dengan mata kita ialah 400 pixel.




```
View[0]:=Box[ObjNo].VertexBank[Box[ObjNo].;
   Polygon[PolyNo].Vertex[1]].X;
View[1]:=Box[ObjNo].VertexBank[Box[ObjNo].;
   Polygon[PolyNo].Vertex[1]].Y;
View[2]:=Box[ObjNo].VertexBank[Box[ObjNo].;
   Polygon[PolyNo].Vertex[1]].Z+400;
```

Ambil titik ketiga dan titik kedua dari poligon sebagai vektor kedua (Vektor B).

```
B[0]:=Box[ObjNo].VertexBank[Box[ObjNo].;
 Polygon[PolyNo].Vertex[2]].X-
      Box[ObjNo].VertexBank[Box[ObjNo].;
        Polygon[PolyNo].Vertex[1]].X;
B[1]:=Box[ObjNo].VertexBank[Box[ObjNo].;
  Polygon[PolyNo].Vertex[2]].Y-
      Box[ObjNo].VertexBank[Box[ObjNo].;
        Polygon[PolyNo].Vertex[1]].Y;
```

```
B[2]:=Box[ObjNo].VertexBank[Box[ObjNo].;
   Polygon[PolyNo].Vertex[2]].Z-
      Box[ObjNo].VertexBank[Box[ObjNo].;
        Polygon[PolyNo].Vertex[1]].Z;
```

Lakukan cross product antara A dan B untuk mendapatkan normal vektor.

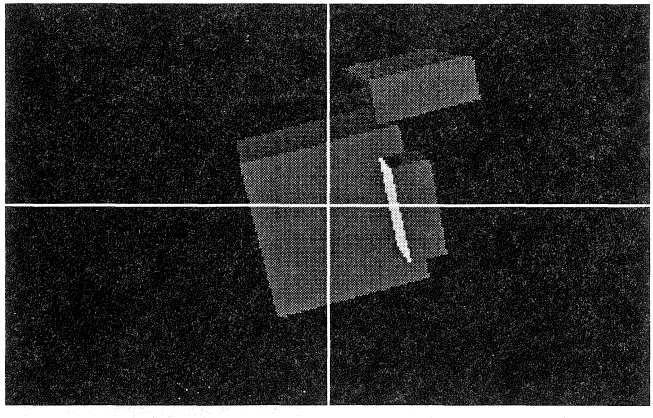
```
Vect_Cross(Normal,A,B);
```

Lakukan Dot Product antara normal vektor dengan garis pandang. Jika hasilnya positif, maka sudut yang dihasilkan ialah sudut acute sehingga poligon membelakangi kita dan harus dicoret dari daftar.

```
If Vect_Dot(Normal,View)>=0 then;
   PolygonList[Ctr].Active:=False;
End;
End;
```

6.3 Vertex Order Filtering

Karena tingkat akurasi dari hasil perhitungan pada komputer tidak terlalu baik pada bilangan yang terlalu besar atau terlalu kecil (overflow terjadi karena tidak cukupnya variabel yang digunakan untuk menampung hasil perhitungan) maka teknik BackFace filtering dengan menggunakan normal vektor tidak terlalu akurat. Cobalah perhatikan sungguh-sungguh pada saat Anda merotasikan object dengan program yang baru saja kita rakit. Kadang-kadang akan terlihat kesalahan dalam menentukan sebuah poligon membelakangi atau menghadap kita.



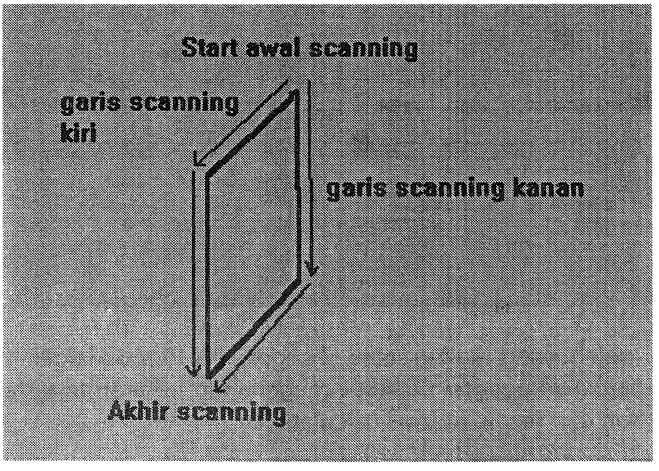
Perhatikan objek yang paling atas. Poligon sisi kirinya tidak digambar, padahal seharusnya poligon tersebut menghadap ke arah kita. Inilah salah satu kesalahan dari teknik normal calculation.

Teknik yang kedua ialah Vertex Order Filtering. Di sini kita tidak perlu melakukan perhitungan dengan cross product, dot product dan lain-lainnya, tetapi cukup membutuhkan sebuah testing yang sangat sederhana dan cukup membutuhkan sebuah If statement. (Pembaca: GRRR!!! Kenapa nggak dijelaskan dari tadi!! Kan nggak perlu repot-repot melakukan proses vektor yang pakai matematika.)

He-he-he ... Bukannya saya mau mempersulit masalah, masalah BackFace filtering dengan perhitungan normal vektor bukannya tidak berguna. Teknik ini nantinya akan menjadi dasar dari perhitungan pencahayaan (light sourcing) yang akan dijelaskan pada bab berikutnya. Percayalah, nggak rugi kok mempelajari teknik tersebut.

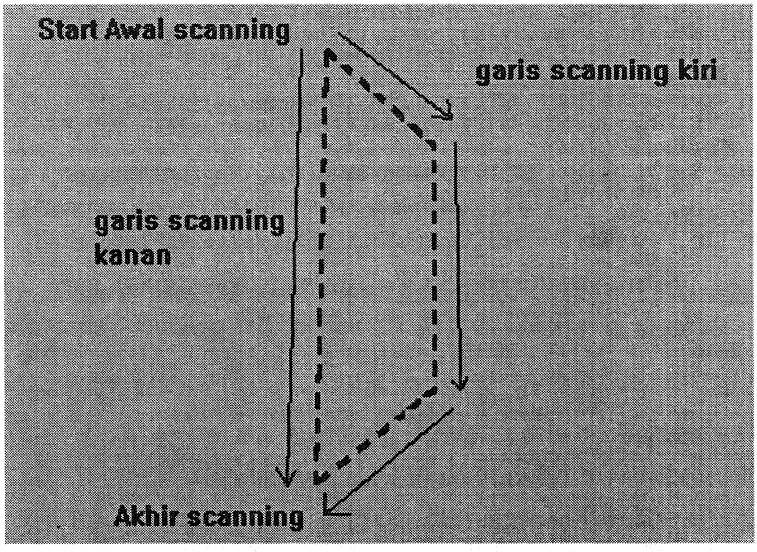
Seperti yang telah kita pelajari pada bab sebelumnya. Pada bagian polygon shading, tentunya pembaca masih ingat pada bagian program di mana dilakukan pendeteksian apakah XL lebih besar dari XR. Jika XL lebih besar maka tukar harga XR dengan XL. XL adalah koordinat horizontal awal dari pengarsiran, sedangkan XR adalah koordinat horizontal akhir dari pengarsiran. XL didapat dari scanning sepanjang garis scanning kiri, sedangkan XR didapat dari scanning sepanjang garis scanning kanan. Jika poligon yang diproyeksikan menghadap ke arah kita, tentu saja aturan susunan titik akan berurutan berlawanan arah jarum jam, sehingga garis scanning kiri benar-benar terjadi pada bagian kiri poligon sedangkan garis scanning kanan pun benar-benar terjadi pada bagian kanan poligon. Jika poligon telah berputar sehingga membelakangi kita, maka garis scanning kiri dilakukan pada bagian kanan dari poligon sedangkan garis scanning kanan dilakukan pada bagian kiri dari poligon. Sehingga XL akan selalu lebih besar dari XR jika poligon membelakangi kita.

Situasi ini dapat digambarkan pada diagram di bawah ini.



Masing-masing garis scanning berada pada tempat yang benar karena pada saat ini poligon menghadap ke arah kita.

Sekarang perhatikan situasi di bawah ini, di mana sebuah poligon membelakangi kita. Mengingat urutan dari titik akan searah jarum jam jika poligon membelakangi kita, maka garis scanning kanan dan kiri akan terbalik posisinya. Sehingga XL yang berjalan sepanjang garis scanning kiri akan lebih besar dari XR yang berjalan sepanjang garis scanning kanan.



Nah, kita dapat memanfaatkan situasi ini. Jika XL lebih besar dari XR maka poligon tersebut membelakangi kita, jadi tidak perlu kita tampilkan. Karena situasi ini diketahui pada saat kita melakukan proses pengarsiran, maka kita tidak perlu melakukan pencoretan poligon dari polygon list melainkan membatalkan pengarsiran poligon yang sedang dilakukan jika XL lebih besar dari XR.

Dalam hal ini, kita cukup mengganti proses di dalam procedure Poly_Shading. Ini adalah procedure Poly_Shading kita yang baru.

```
Procedure Poly_Shading(Poly:Poly_Rec;ObjNo:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
```

```
  YMinIndex    : Integer;
  YMaxIndex    : Integer;
  RightCtr     : Integer;
  LeftCtr      : Integer;
  XL,XR,Bf     : Longint;
  Dx,Dy        : Real;
  Finish       : Boolean;
  Ty           : Longint;
Begin
  YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
  YMinIndex:=0;
  YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
  YMaxIndex:=0;

  For A:=1 to Poly.TotalVertex-1 do
    Begin
      Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
      If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
        Begin
          YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
          YMinIndex:=A;
        End;
      If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
        Begin
          YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
          YMaxIndex:=A;
        End;
    End;

  LeftStart:=YMinIndex; RightStart:=YMinIndex;
  YCtr :=YMin;
  LeftEnd:=LeftStart+1;
  If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
  RightEnd:=RightStart-1;
  If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
  Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
     Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
  Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
     Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
  Finish:=False;
  While (Dy=0) and (Not Finish) do
    Begin
      LeftStart:=LeftStart+1;
      If LeftStart>=Poly.TotalVertex then LeftStart:=0;
      LeftEnd:=LeftStart+1;
      Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
        Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
      Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
        Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
      If LeftStart=YMaxIndex then Finish:=True;
    End;
  If Not Finish then
    Begin
      LeftSlope:=(Dx)/(Dy);
      Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
        Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
      Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
        Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
      While (Dy=0) and (Not Finish) do
        Begin
```

```
      RightStart:=RightStart-1;
      If RightStart<0 then RightStart:=Poly.TotalVertex-1;
      RightEnd:=RightStart-1;
      Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
         Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
      Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
         Yp-Box[ObjNo].VertexBank;
          [Poly.Vertex[RightStart]].Yp+1;
      If RightStart=YMaxIndex then Finish:=True;
    End;
  RightSlope:=(Dx)/(Dy);
  XLeft :=Box[ObjNo].VertexBank;
   [Poly.Vertex[LeftStart]].Xp+LeftSlope;
  XRight:=Box[ObjNo].VertexBank;
   [Poly.Vertex[RightStart]].Xp+RightSlope;
  While (Not Finish) do
    Begin
      XL:=Round(XLeft);
      XR:=Round(XRight);
```

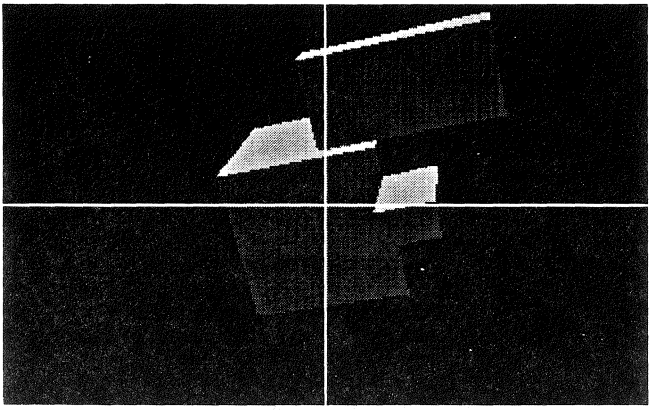
Lakukan pengecekan, jika XL lebih besar dari XR maka jangan lakukan shading karena poligon tersebut membelakangi kita.

```
      If XL>XR then Finish:=True;
```

Selebihnya sama persis dengan Poly_Shading kita yang lama, hanya tidak adanya proses penukaran XL dengan XR, jika XL lebih besar dari XR.

```
      If XL<0 then XL:=0;
      If XR>319 then XR:=319;
      If (YCtr>=0) and (Not Finish) then
       If XL=XR then PutPixel(Buff^,XL,YCtr,Poly.Color) else
        If XL<XR then
          HLines(Buff^,XL,XR,YCtr,Poly.Color);
      XLeft:=XLeft+LeftSlope;
      XRight:=XRight+RightSlope;
      Inc(YCtr);
      If YCtr>199 then Finish:=True;
      If Not Finish then
       If (YCtr>Box[ObjNo].VertexBank;
          [Poly.Vertex[LeftEnd]].Yp) then
        Begin
          Repeat
            LeftStart:=LeftStart+1;
            If LeftStart>=Poly.TotalVertex then LeftStart:=0;
            LeftEnd:=LeftStart+1;
            If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
            Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                Xp-Box[ObjNo].VertexBank;
                 [Poly.Vertex[LeftStart]].Xp;
            Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                Yp-Box[ObjNo].VertexBank;
                 [Poly.Vertex[LeftStart]].Yp+1;
            If LeftStart=YMaxIndex then Finish:=True;
          Until (Dy<>0) or Finish;
          If Not Finish then
            Begin
              LeftSlope:=(Dx)/(Dy);
              XLeft:=Box[ObjNo].VertexBank;
                 [Poly.Vertex[LeftStart]].Xp+LeftSlope;
            End;
```

```
            End;
          If Not Finish then
            Begin
              If YCtr>Box[ObjNo].VertexBank;
                [Poly.Vertex[RightEnd]].Yp then
                Begin
                  Repeat
                    RightStart:=RightStart-1;
                    If RightStart<0 then;
                      RightStart:=Poly.TotalVertex-1;
                    RightEnd:=RightStart-1;
                    If RightEnd<0 then;
                      RightEnd:=Poly.TotalVertex-1;
                    Dx:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[RightEnd]].;
                        Xp-Box[ObjNo].VertexBank;
                          [Poly.Vertex[RightStart]].Xp;
                    Dy:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[RightEnd]].;
                        Yp-Box[ObjNo].VertexBank;
                          [Poly.Vertex[RightStart]].Yp+1;
                    If RightStart=YMaxIndex then Finish:=True;
                  Until (Dy<>0) or Finish;
                  If Not Finish then
                    Begin
                      RightSlope:=(Dx)/(Dy);
                      XRight:=Box[ObjNo].VertexBank;
                        [Poly.Vertex[RightStart]].Xp;
                      XRight:=XRight+RightSlope;
                    End;
                End;
            End;
        End;
    End;
End;
```

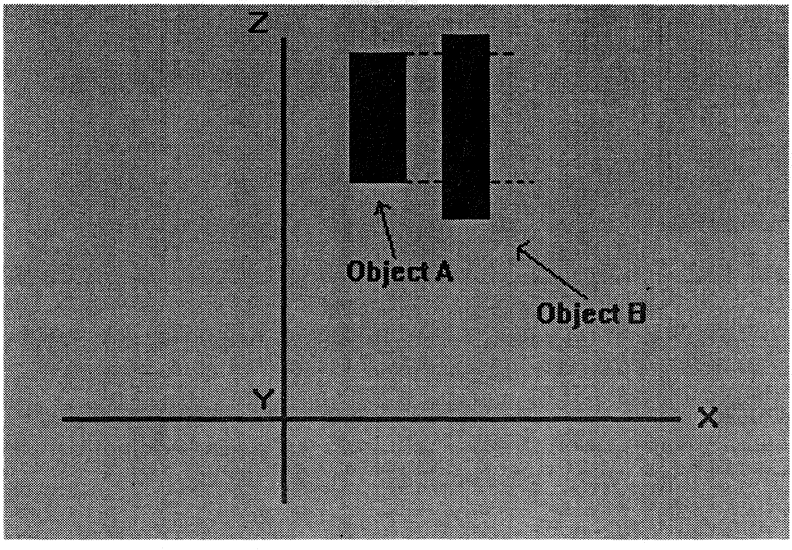


Nah, dengan teknik yang baru, kesalahan tidak akan terjadi.

6.4 Depth Sorting

Tentunya jika Anda perhatikan baik-baik, program kita yang terakhir masih ada kesalahannya. Kesalahan ini terletak pada urutan penggambaran dari objek. Perhatikan objek yang paling kecil. Objek tersebut berada pada koordinat Z terjauh, seharusnya dia berada di belakang objek yang paling besar. Tetapi karena tidak adanya urutan dalam penggambaran poligon, maka kadang-kadang terjadi kesalahan urutan penggambaran. Sudah selayaknya kita melakukan penggambaran dari objek yang terjauh ke objek yang terdekat sehingga secara otomatis, objek yang terjauh akan tertumpuk oleh objek yang lebih dekat. Pada program kita yang terakhir, kita masih belum menggunakan teknik ini. Teknik mengurutkan objek ini disebut depth sorting atau PAINTER SORT ALGORITHM. Sebenarnya ini bukan merupakan jenis filter. Tetapi karena penggunaanya akan berkenaan dengan polygon list, maka kita golongkan saja sebagai salah satu dari filter.

Untuk menambah akurasi dari sorting, dilakukan sorting per-poligon dalam polygon list. Karena sorting per-objek tidak dapat menjamin urutan penggambaran secara akurat. Ini terjadi karena sebuah objek tidak hanya sebesar titik, tetapi mencakup sebuah area. Perhatikan diagram di bawah ini.

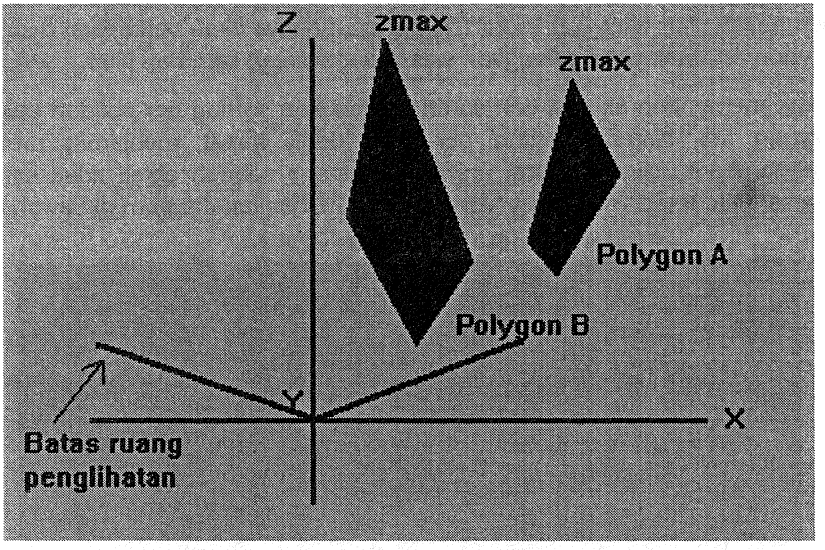


Ini adalah objek yang dilihat dari sudut pandang atas. Perhatikan titik dari objek A yang memiliki koordinat Z terbesar, ternyata lebih kecil dari titik dari objek B yang memiliki koordinat Z terbesar. Tetapi titik yang berkoordinat Z terkecil dari objek B ternyata lebih jauh dari titik yang berkoordinat Z terkecil

dari objek A. Berdasarkan urutan titik berkoordinat Z terbesar, maka objek B lebih jauh sehingga B harus digambar lebih dahulu, tetapi menurut urutan titik berkoordinat Z terkecil, objek A lebih jauh sehingga A harus digambar lebih dahulu. Inilah yang menjadi masalah. Untuk itu dilakukan penggambaran per-poligon.

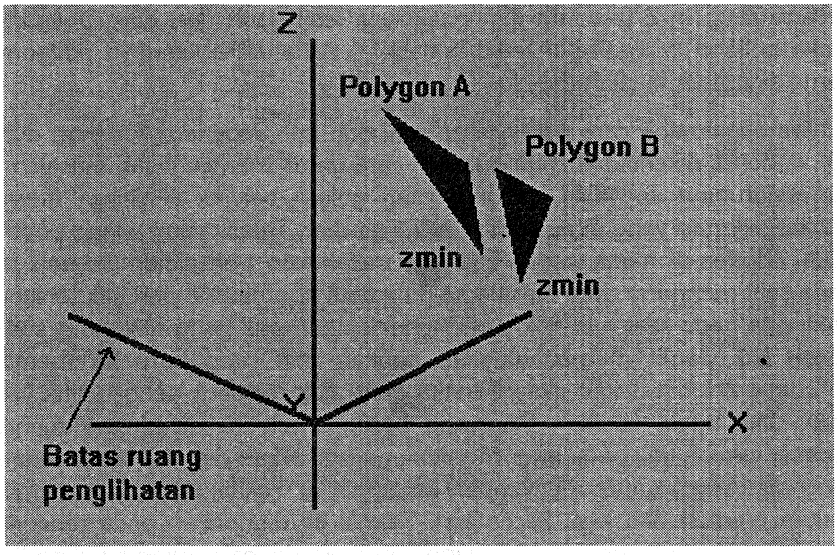
Tetapi jangan keburu senang karena poligon pun dibentuk dari beberapa titik, sehingga memiliki masalah yang serupa. Titik yang mana yang akan kita gunakan sebagai patokan untuk proses sorting?

Dalam hal ini kita dihadapkan pada tiga buah pilihan. Pertama ialah sorting berdasarkan titik yang memiliki koordinat Z terbesar. Dalam hal ini sorting akan mengalami kesalahan jika dihadapkan pada situasi berikut.



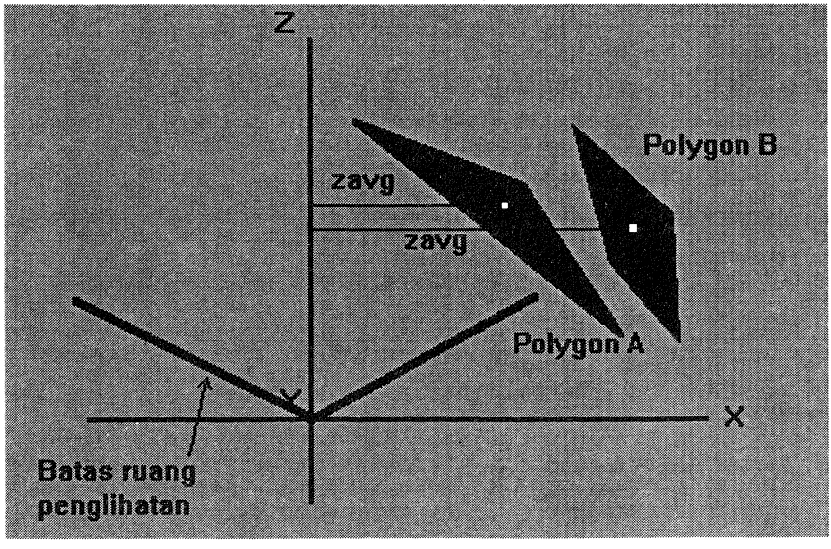
Seharusnya poligon A digambar terlebih dahulu, kemudian baru poligon B, tetapi gara-gara Zmax dari poligon A lebih dekat dari Zmax poligon B, maka poligon B digambar terlebih dahulu.

Jika kita melakukan sort berdasarkan Zmin, juga ada kemungkinan terjadi kesalahan seperti yang ditunjukkan pada diagram berikut.



Di sini terlihat, Zmin dari poligon A lebih jauh dari Zmin poligon B sehingga poligon A digambar terlebih dahulu. Secara kenyataan, seharusnya poligon B digambar terlebih dahulu.

Nah, pilihan terakhir ialah melakukan sorting berdasarkan rata-rata antara Zmax dan Zmin. Hal ini pun ternyata dapat juga mengakibatkan kesalahan. Perhatikan gambar di bawah ini.



Karena Zavg = Z rata-rata dari poligon A lebih jauh dari Zavg poligon B maka poligon A digambar terlebih dahulu. Padahal sebenarnya poligon B harus digambar terlebih dahulu.

Lantas bagaimana cara pemecahan masalah ini? Sebenarnya dalam dunia tiga dimensi, dikenal beberapa macam cara untuk memecahkan masalah ini dan seluruhnya memiliki kalkulasi yang cukup banyak sehingga menyebabkan animasi yang dihasilkan menjadi lambat. Teknik yang paling populer ialah ZBuffer di mana setiap kali kita melakukan peletakkan sebuah titik pada saat mengarsir, kita masukkan pula data koordinat Z dari titik tersebut. Sehingga pada saat ada titik yang hendak ditumpuk pada titik yang sudah berisi, kita dapat membandingkan apakah titik yang baru ini memiliki koordinat Z lebih dekat, jika ya maka ganti titik tersebut dengan titik yang baru. Jika tidak, abaikan titik yang baru. Untuk itu diperlukan memori sebesar jumlah titik pada layar monitor yang dikalikan dengan besar variabel yang digunakan untuk menyimpan koordinat Z. Teknik ini sangat memboroskan memori dan juga kecepatan. Teknik ini digunakan pada rendering process dari 3D Studio. Teknik yang lain ialah dengan menggunakan lima tingkat polygon testing. Teknik ini tidak membutuhkan banyak memori tetapi cukup banyak mempengaruhi kecepatan dari animasi.

Dalam edisi kita kali ini, kedua teknik tersebut tidak akan kita kita bahas. Kita akan tetap bertahan memakai teknik sorting biasa dengan memilih salah satu dari tiga pilihan (Zmax, Zmin atau Zavg). Dalam hal ini kita pilih saja Zmin.

Perhatikan modifikasi dari B3SHADE4 di bawah ini.

```
{-----------------------------------------------                }
{ Program     : B3SHADE5.PAS                                   }
{ Coder       : Agustinus Nalwan                               }
{ Description: Program transformasi object                     }
{               dengan filter Painter Sort                     }
{-----------------------------------------------                }

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
       Xp,Yp   : Longint;
     End;
```

```
    ObjRect = Record
      Color      : Byte;
      VertexBank : Array[0..20] of Vertex_Rec;
      Polygon    : Array[0..20] of Poly_Rec;
      TotalVertex: Integer;
      TotalPoly  : Integer;
      X,Y,Z      : Real;
    End;

    PolyList_Rec = Record
      ObjNo,PolyNo : Integer;
      Active       : Boolean;
      ZMin         : Real;
      ZMax         : Real;
      ZAvg         : Real;
    End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ListCtr:Integer;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=-lx;Box[No].VertexBank[0].Y:=ly;
       Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;Box[No].VertexBank[1].Y:=ly;
       Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
       Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
       Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;Box[No].VertexBank[4].Y:=ly;
       Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
       Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
       Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
       Box[No].VertexBank[7].Z:=lz;
    Box[No].Polygon[0].TotalVertex:=4;
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;
```

```
    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;

    Box[No].Polygon[0].Color:=17;
    Box[No].Polygon[1].Color:=19;
    Box[No].Polygon[2].Color:=21;
    Box[No].Polygon[3].Color:=23;
    Box[No].Polygon[4].Color:=25;
    Box[No].Polygon[5].Color:=27;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,90,0,850,20,50,30,11);
    Create_Box(2,80,130,850,50,20,100,12);
```

```
    Remove_PolyList;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
            PolygonList[ListCtr].ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr].ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then;
                   PolygonList[ListCtr].ZMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr].ZMin>VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z then
                   PolygonList[ListCtr].ZMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;
              End;
            PolygonList[ListCtr].ZAvg:=(PolygonList[ListCtr].;
               ZMax+PolygonList[ListCtr].ZMin)/2;
            Inc(ListCtr);
          End;
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
```

```
        PolyNo:=PolygonList[A].PolyNo;
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;        Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
    LeftCtr     : Integer;
    XL,XR,Bf    : Longint;
    Dx,Dy       : Real;
    Finish      : Boolean;
    Ty          : Longint;
  Begin
    YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMinIndex:=0;
    YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMaxIndex:=0;

    For A:=1 to Poly.TotalVertex-1 do
      Begin
        Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMinIndex:=A;
          End;
        If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMaxIndex:=A;
          End;
      End;

    LeftStart:=YMinIndex; RightStart:=YMinIndex;
```

```
    YCtr :=YMin;
    LeftEnd:=LeftStart+1;
    If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
    RightEnd:=RightStart-1;
    If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
    Finish:=False;
    While (Dy=0) and (Not Finish) do
      Begin
        LeftStart:=LeftStart+1;
        If LeftStart>=Poly.TotalVertex then LeftStart:=0;
        LeftEnd:=LeftStart+1;
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
        If LeftStart=YMaxIndex then Finish:=True;
      End;
    If Not Finish then
      Begin
        LeftSlope:=(Dx)/(Dy);
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
        While (Dy=0) and (Not Finish) do
          Begin
            RightStart:=RightStart-1;
            If RightStart<0 then RightStart:=Poly.TotalVertex-1;
            RightEnd:=RightStart-1;
            Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Xp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
            Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].Yp-
Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
            If RightStart=YMaxIndex then Finish:=True;
          End;
        RightSlope:=(Dx)/(Dy);
        XLeft
:=Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp+LeftSlope;

XRight:=Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp+RightSlope
;
        While (Not Finish) do
          Begin
            XL:=Round(XLeft);
            XR:=Round(XRight);

            If XL>XR then Finish:=True;

            If XL<0 then XL:=0;
            If XR>319 then XR:=319;
            If (YCtr>=0) and (Not Finish) then
             If XL=XR then PutPixel(Buff^,XL,YCtr,Poly.Color) else
              If XL<XR then
                HLines(Buff^,XL,XR,YCtr,Poly.Color);
            XLeft:=XLeft+LeftSlope;
            XRight:=XRight+RightSlope;
            Inc(YCtr);
```

```
            If YCtr>199 then Finish:=True;
            If Not Finish then
             If (YCtr>Box[ObjNo].VertexBank;
               [Poly.Vertex[LeftEnd]].Yp) then
              Begin
                Repeat
                  LeftStart:=LeftStart+1;
                  If LeftStart>=Poly.TotalVertex then LeftStart:=0;
                  LeftEnd:=LeftStart+1;
                  If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
                  Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                       Xp-Box[ObjNo].VertexBank;
                        [Poly.Vertex[LeftStart]].Xp;
                  Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                       Yp-Box[ObjNo].VertexBank;
                        [Poly.Vertex[LeftStart]].Yp+1;
                  If LeftStart=YMaxIndex then Finish:=True;
                Until (Dy<>0) or Finish;
                If Not Finish then
                  Begin
                    LeftSlope:=(Dx)/(Dy);
                    XLeft:=Box[ObjNo].VertexBank;
                       [Poly.Vertex[LeftStart]].Xp+LeftSlope;
                  End;
              End;
            If Not Finish then
              Begin
                If YCtr>Box[ObjNo].VertexBank;
                  [Poly.Vertex[RightEnd]].Yp then
                  Begin
                    Repeat
                      RightStart:=RightStart-1;
                      If RightStart<0 then;
                       RightStart:=Poly.TotalVertex-1;
                      RightEnd:=RightStart-1;
                      If RightEnd<0 then;
                       RightEnd:=Poly.TotalVertex-1;
                      Dx:=Box[ObjNo].VertexBank;
                       [Poly.Vertex[RightEnd]].;
                         Xp-Box[ObjNo].VertexBank;
                           [Poly.Vertex[RightStart]].Xp;
                      Dy:=Box[ObjNo].VertexBank;
                       [Poly.Vertex[RightEnd]].;
                         Yp-Box[ObjNo].VertexBank;
                           [Poly.Vertex[RightStart]].Yp+1;
                      If RightStart=YMaxIndex then Finish:=True;
                    Until (Dy<>0) or Finish;
                    If Not Finish then
                      Begin
                        RightSlope:=(Dx)/(Dy);
                        XRight:=Box[ObjNo].VertexBank;
                             [Poly.Vertex[RightStart]].Xp;
                        XRight:=XRight+RightSlope;
                      End;
                  End;
              End;
          End;
      End;
  End;

Procedure Display_PList_Shading;
```

```
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        Poly_Shading(Box[ObjNo].Polygon[PolyNo],ObjNo);
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;
```

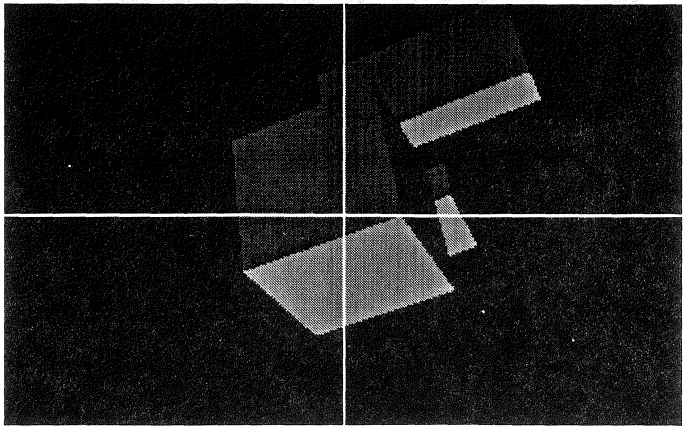
```
Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        Box[A].X:=Box[A].X+Tx;
        Box[A].Y:=Box[A].Y+Ty;
        Box[A].Z:=Box[A].Z+Tz;
      End;
  End;

Procedure Painter_Sort;
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
    For A:=0 to ListCtr-2 do
     If PolygonList[A].active then
      For B:=A+1 to ListCtr-1 do
        If PolygonList[B].active then
          If PolygonList[A].ZMin<PolygonList[B].ZMin then
            Begin
              Bf:=PolygonList[A];
              PolygonList[A]:=PolygonList[B];
              PolygonList[B]:=Bf;
            End;
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Ch:=' ';
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Object;
        Painter_Sort;
        Display_PList_Shading;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
```

```
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          'r':Scale(1.1,1,1);
          'f':Scale(0.9,1,1);
          't':Scale(1,1.1,1);
          'g':Scale(1,0.9,1);
          'y':Scale(1,1,1.1);
          'h':Scale(1,1,0.9);
          '7':Dilatasi(10,0,0);
          '1':Dilatasi(-10,0,0);
          '8':Dilatasi(0,10,0);
          '2':Dilatasi(0,-10,0);
          '9':Dilatasi(0,0,10);
          '3':Dilatasi(0,0,-10);
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

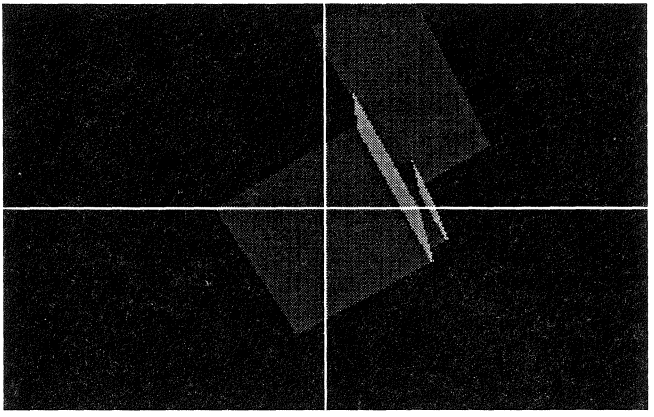
PENJELASAN PROGRAM

Jika Anda jalankan program di atas, akan terlihat hasil proyeksi obj k kali ini ditampilkan dengan urutan yang benar (hampir benar).

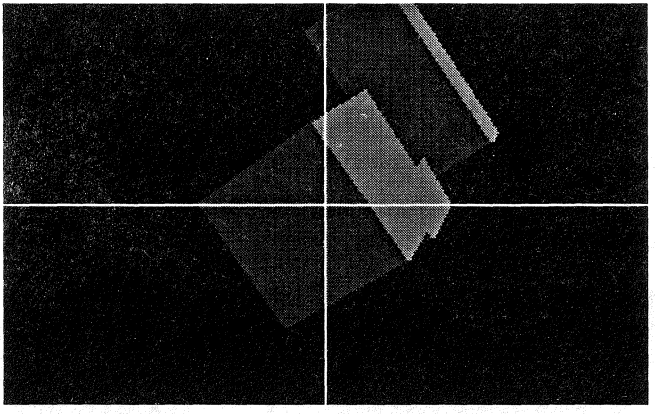
Coba lakukan rotasi dan dilatasi. Memang seperti yang telah kita bahas, pada beberapa saat terjadi kesalahan sorting.



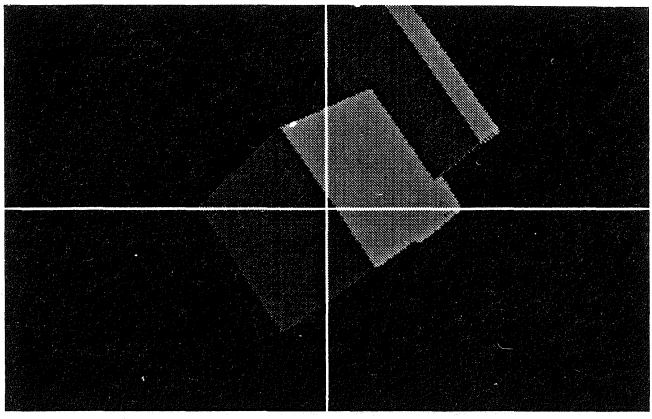
Poligon tersorting dengan benar. Sekarang objek yang paling kecil telah berada pada posisi yang benar yaitu di belakang objek yang terbesar karena koordinat Z-nya lebih jauh.



Kesalahan masih dapat terjadi. Perhatikan gambar di atas ini. Objek yang paling kecil berada di tengah kedua objek.



Setelah dilakukan rotasi, objek tersebut masih pada posisi di tengah kedua objek.



Setelah rotasi lebih lanjut, objek kecil tersebut sekarang berada di belakang kedua objek. Ops!!! kesalahan terjadi. Ini terjadi karena kita memakai patokan ZMin sebagai kunci dari proses sort.

PENJELASAN LISTING PROGRAM

Di sini dilakukan beberapa modifikasi dari program B3SHADE4.PAS kita yang terdahulu.

Pada Polygon List data structure ditambahkan ZMin, ZMax dan ZAvg untuk menyimpan koordinat Z terkecil, terbesar dan rata-rata dari setiap poligonnya.

```
PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
       ZMin         : Real;
       ZMax         : Real;
       ZAvg         : Real;
     End;
```

Kemudian pada proses pendaftaran poligon ke dalam Polygon List, dilakukan pencarian Zmax, Zmin dan Zavg dari setiap poligon.

```
Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
```

Lakukan scanning untuk setiap poligon dalam polygon list.

```
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
```

```
PolygonList[ListCtr].PolyNo:=A;
PolygonList[ListCtr].Active:=True;
```

Asumsikan ZMax dan ZMin adalah pada titik pertama dari poligon yang sedang diakses.

```
PolygonList[ListCtr].ZMax:=VertexBank;
   [Polygon[A].Vertex[0]].Z+Z;
PolygonList[ListCtr].ZMin:=VertexBank;
   [Polygon[A].Vertex[0]].Z+Z;
```

Lakukan scanning pada setiap titik dari Polygon List yang sedang diakses.

```
For B:=0 to Polygon[A].TotalVertex-1 do
  Begin
```

Jika ditemukan ZMax dan ZMin baru, update harga yang lama.

```
If PolygonList[ListCtr].ZMax<VertexBank;
  [Polygon[A].Vertex[B]].Z+Z then
    PolygonList[ListCtr].ZMax:=VertexBank;
       [Polygon[A].Vertex[B]].Z+Z;
If PolygonList[ListCtr].ZMin>VertexBank;
       [Polygon[A].Vertex[B]].Z+Z then
    PolygonList[ListCtr].ZMin:=VertexBank;
       [Polygon[A].Vertex[B]].Z+Z;
End;
```

Cari ZAvg dengan membagi hasil penjumlahan dari ZMax dan ZMin.

```
PolygonList[ListCtr].ZAvg:=(PolygonList[ListCtr].;
    ZMax+PolygonList[ListCtr].ZMin)/2;
  Inc(ListCtr);
End;
  End;
End;
```

Langkah selanjutnya ialah penambahan fungsi sorting.

```
Procedure Painter_Sort;
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
```

Lakukan bubble sort untuk mengurutkan poligon berdasarkan ZMin. Sebenarnya di sini bisa dilakukan proses optimasi kecepatan dengan menggunakan teknik sorting yang lebih bagus seperti Quick Sort. Tetapi jika jumlah poligon sedikit, penggunaan Quick Sort malahan akan memperlambat animasi karena Quick Sort memiliki performance yang jelek jika jumlah element yang akan disorting sedikit.

```
For A:=0 to ListCtr-2 do
 If PolygonList[A].active then
  For B:=A+1 to ListCtr-1 do
    If PolygonList[B].active then
```

Jika ditemukan ZMin yang lebih kecil, tukar poligon A dengan poligon B.

```
        If PolygonList[A].ZMin<PolygonList[B].ZMin then
          Begin
            Bf:=PolygonList[A];
            PolygonList[A]:=PolygonList[B];
            PolygonList[B]:=Bf;
          End;
End;
```

Nah, modifikasi terakhir ialah pada main program. Kita harus pandai-pandainya menempatkan fungsi sorting ini sehingga diperoleh kecepatan yang paling maksimal. Untuk itu jangan tempatkan Painter_Sort ini sebelum proses BackFace Filter jika Anda menggunakan teknik BackFace Filter yang menggunakan kalkulasi garis normal. Karena dengan jumlah element yang semakin banyak pada polygon list, sort yang akan dilakukan akan semakin lambat. Tempatkan Painter_Sort di mana polygon list telah di-minimize seminim mungkin. Karena di sini kita tidak menggunakan BackFace Filter procedure maka kita hanya dapat menempatkannya sebelum penggambaran poligon.

```
Procedure Main;
Var Ch:Char;
  Begin
    Ch:=' ';
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Object;
```

Panggil Painter_Sort sebelum dilakukan proses penggambaran poligon.

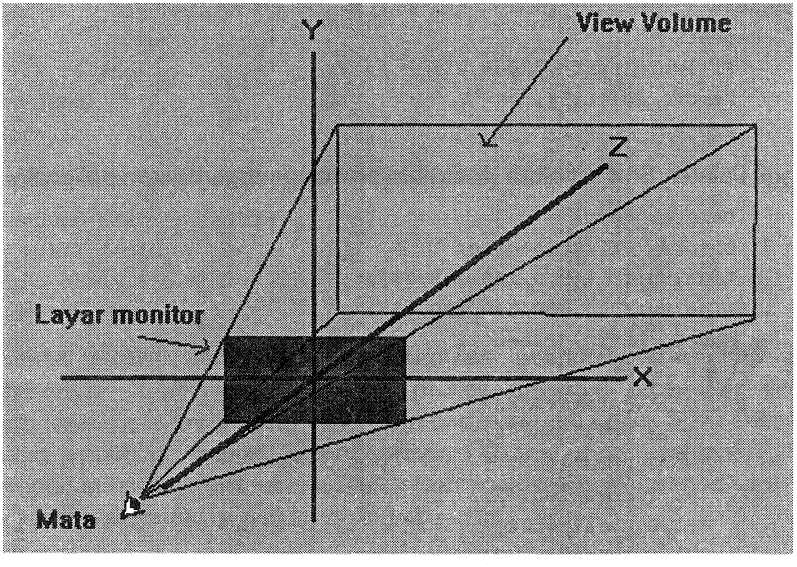
```
        Painter_Sort;
        Display_PList_Shading;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          'r':Scale(1.1,1,1);
          'f':Scale(0.9,1,1);
          't':Scale(1,1.1,1);
```

```
          'g':Scale(1,0.9,1);
          'y':Scale(1,1,1.1);
          'h':Scale(1,1,0.9);
          '7':Dilatasi(10,0,0);
          '1':Dilatasi(-10,0,0);
          '8':Dilatasi(0,10,0);
          '2':Dilatasi(0,-10,0);
          '9':Dilatasi(0,0,10);
          '3':Dilatasi(0,0,-10);
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
End;
```

6.5 View Volume Filtering

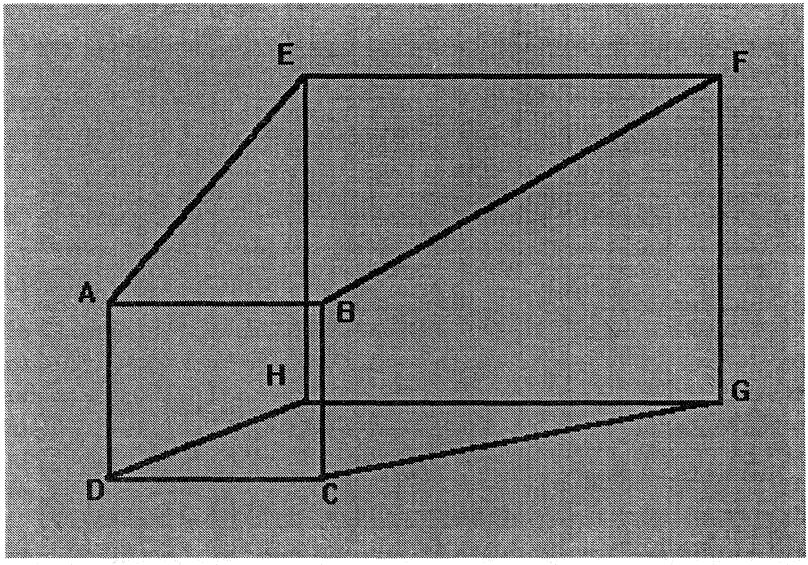
Filter yang ketiga ialah Volume Filter. Di sini dilakukan pencoretan poligon-poligon dalam polygon list yang berada di luar view volume (ruang penglihatan kita).

Perhatikan diagram berikut.

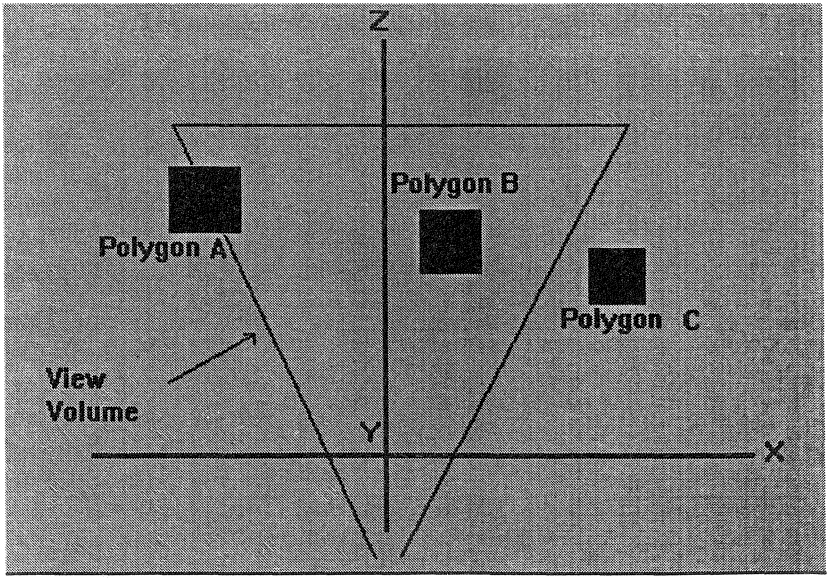


Ini adalah ruang penglihatan kita (View Volume) yang berbentuk piramida yang terpotong oleh layar monitor. Hanya di dalam area piramida inilah seluruh objek di dunia tiga dimensi dapat terlihat.

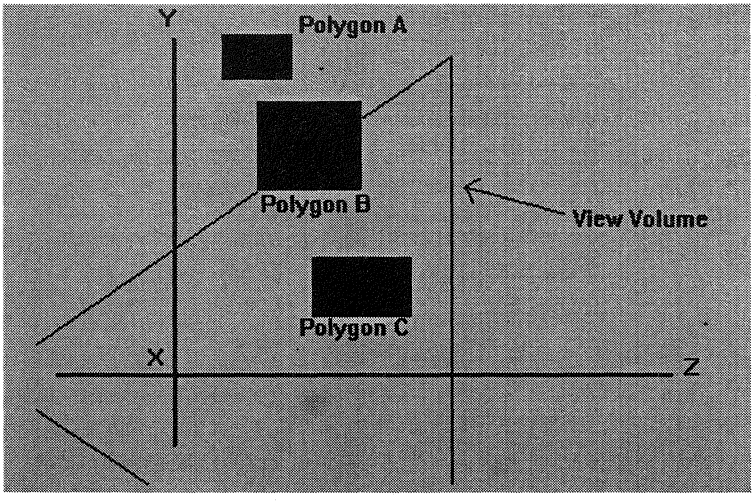
Karena piramida ini terpotong dan memiliki 6 sisi yaitu sisi yang berdasarkan gambar di bawah ini ialah sisi ABCD, EFGH sebagai pembatas terhadap sumbu Z, sisi ADHE dan BCGF sebagai pembatas berdasarkan sumbu X dan sisi AEFB dan DCGH sebagai pembatas berdasarkan sumbu Y. Untuk itu kita akan membagi sistem pembatasan ini menjadi tiga bagian.



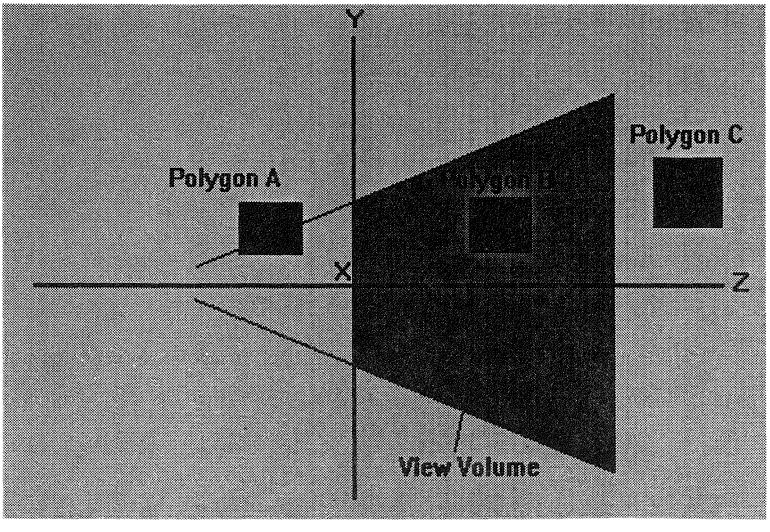
View Volume yang dapat dibayangkan sebagai piramida yang terpotong.



Bagian pertama ialah pembatasan pada bidang sumbu X . Di sini dilakukan pendeteksian apakah sebuah poligon berada di luar, di dalam atau terpotong batas sumbu X. Pada situasi di atas, poligon A terpotong oleh batas sumbu X. Poligon B berada di dalam batas sumbu X, sedangkan poligon C berada di batas sumbu X. Jika poligon berada di luar batas sumbu X, maka poligon dicoret dari daftar polygon list.



Bagian kedua ialah pengecekan terhadap batas Y. Jika poligon berada di luar batas Y maka poligon dicoret dari daftar polygon list. Pada situasi di atas, poligon A berada di luar batas sumbu Y, poligon B terpotong batas sumbu Y sedangkan poligon C berada di dalam batas sumbu Y.



Bagian ketiga ialah pembatasan Z. Pembatasan Z dilakukan dengan memotong ruang penglihatan pada koordinat 0, juga pada suatu tempat yang ditentukan sebagai koordinat Z terbesar, di mana mata kita masih mampu melihat sebuah objek. Biasanya pembatasan Z ini dilakukan untuk mengurangi jumlah poligon yang perlu ditampilkan pada layar monitor. Karena itulah pada beberapa game tiga dimensi, objek yang terlalu jauh tidak terlihat. Begitu dekat, baru objek ini tiba-tiba muncul. Teknik ini dapat dilihat pada game DAYTONA buatan SEGA. Perhatikan tebing-tebing karang yang membentuk arena pertandingan. Pada jarak jauh, tidak seluruh bagian terlihat, tetapi begitu kita mendekat, poligon-poligon tersebut mulai bermunculan. Pada situasi di atas, poligon A dan C berada di luar batas Z sedangkan B berada di dalam.

Nah, kita telah mempelajari bagaimana algorithm dari View Volume Filter. Sekarang yang menjadi masalah ialah bagaimana cara melakukan pendeteksian secara programming?

Pada pembatasan Z, sangatlah mudah. Karena bidang pembatasan kiri tegak lurus terhadap sumbu Z dan berada pada titik origin. Kita dapat mendeteksi apakah koordinat ZMax (koordinat Z paling besar yang dimiliki poligon) dari poligon negatif atau positif. Jika negatif maka poligon berada di luar ruang penglihatan. Kemudian untuk bidang pembatasan kanan, karena bidang ini juga tegak lurus terhadap sumbu Z, kita tinggal mendeteksi apakah ZMin (koordinat Z paling kecil yang dimiliki poligon) dari poligon lebih besar dari batas kanan Z. Jika lebih besar maka poligon berada di luar ruang penglihatan.

Karena pembatasan Z ini tidak memerlukan perhitungan yang rumit dan memakan kecepatan komputer, maka pembatasan Z ini dilakukan pertama kali. Jika pembatasan ini menunjukkan bahwa poligon di luar bidang penglihatan, kita tidak perlu lagi melakukan tes-tes yang lain.

Sekarang pembatasan sumbu X dan Y. Karena kedua bidang pembatasnya tidak tegak lurus terhadap salah satu sumbu, maka perhitungan pembatasan secara tiga dimensi sulit dicapai. Karena kita harus mengetahui harga batas X dan Y pada suatu koordinat Z (mengingat batas X dan Y bertambah terus jika koordinat Z juga semakin bertambah).

Dengan akal licik, kita balik saja prosesnya!!! Jika poligon di luar ruang pembatasan, pasti pada saat dilakukan proses proyeksi, poligon hasil proyeksi pasti terletak di luar layar monitor. Nah kalau begitu pengecekan ini dapat kita lakukan setelah poligon diproyeksikan atau dengan kata lain dilakukan pengecekan secara dua dimensi. Tentunya hal ini sangat mudah dilakukan.

Ok, sekarang kita akan memodifikasi program kita sebelumnya sehingga

dapat menggunakan View Volume Filter.

```
{-----------------------------------------------          }
{ Program     : B3SHADE6.PAS                             }
{ Coder       : Agustinus Nalwan                         }
{ Description: Program transformasi object               }
{               dengan filter View Volume                }
{-----------------------------------------------          }

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
       Xp,Yp   : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
       X,Y,Z      : Real;
     End;

     PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
       ZMin         : Real;
       ZMax         : Real;
       XMin         : Real;
       XMax         : Real;
       YMin         : Real;
       YMax         : Real;
       Color        : Integer;
     End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ListCtr:Integer;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;
```

```
Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=-lx;Box[No].VertexBank[0].Y:=ly;
        Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;Box[No].VertexBank[1].Y:=ly;
        Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
        Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
        Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;Box[No].VertexBank[4].Y:=ly;
        Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
        Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
        Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
        Box[No].VertexBank[7].Z:=lz;
    Box[No].Polygon[0].TotalVertex:=4;
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;

    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
```

```
    Box[No].Polygon[5].Vertex[3]:=6;

    Box[No].Polygon[0].Color:=10;
    Box[No].Polygon[1].Color:=20;
    Box[No].Polygon[2].Color:=30;
    Box[No].Polygon[3].Color:=40;
    Box[No].Polygon[4].Color:=50;
    Box[No].Polygon[5].Color:=60;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,90,0,850,20,50,30,11);
    Create_Box(2,80,130,850,50,20,100,12);
    Remove_PolyList;
    For A:=1 to 91 do
      Begin
        SetVisualPalette(A,63-Round(A/1.5),63-Round(A/1.5),;
          63-Round(A/1.5));
      End;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
```

```
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
            PolygonList[ListCtr].ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].XMax:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr].XMin:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr].YMax:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            PolygonList[ListCtr].YMin:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr].ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr].ZMin>VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;

                If PolygonList[ListCtr].YMax<VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                    PolygonList[ListCtr].YMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr].YMin>VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                    PolygonList[ListCtr].YMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr].XMax<VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                    PolygonList[ListCtr].XMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Xp;
                If PolygonList[ListCtr].XMin>VertexBank;
                        [Polygon[A].Vertex[B]].Xp then
                    PolygonList[ListCtr].XMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Xp;
              End;
            Inc(ListCtr);
          End;
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
```

```
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;         Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
    LeftCtr     : Integer;
    XL,XR,Bf    : Longint;
    Dx,Dy       : Real;
    Finish      : Boolean;
    Ty          : Longint;
  Begin
    YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMinIndex:=0;
    YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMaxIndex:=0;

    For A:=1 to Poly.TotalVertex-1 do
      Begin
        Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMinIndex:=A;
          End;
        If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMaxIndex:=A;
          End;
      End;

    LeftStart:=YMinIndex; RightStart:=YMinIndex;
    YCtr :=YMin;
```

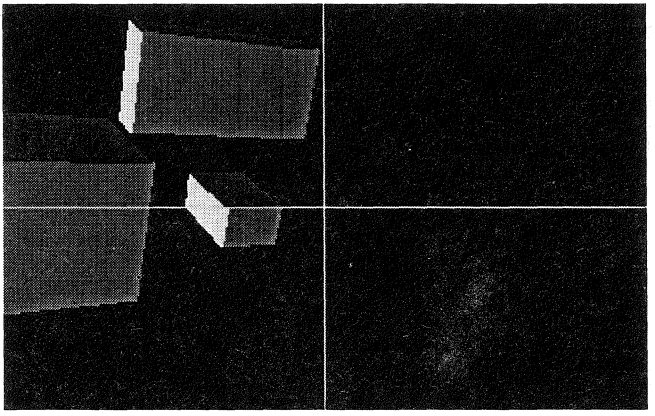
```
LeftEnd:=LeftStart+1;
If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
RightEnd:=RightStart-1;
If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
    Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
    Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
Finish:=False;
While (Dy=0) and (Not Finish) do
  Begin
    LeftStart:=LeftStart+1;
    If LeftStart>=Poly.TotalVertex then LeftStart:=0;
    LeftEnd:=LeftStart+1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
       Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
       Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
    If LeftStart=YMaxIndex then Finish:=True;
  End;
If Not Finish then
  Begin
    LeftSlope:=(Dx)/(Dy);
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
       Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
       Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
    While (Dy=0) and (Not Finish) do
      Begin
        RightStart:=RightStart-1;
        If RightStart<0 then RightStart:=Poly.TotalVertex-1;
        RightEnd:=RightStart-1;
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
           Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
           Yp-Box[ObjNo].VertexBank;
            [Poly.Vertex[RightStart]].Yp+1;
        If RightStart=YMaxIndex then Finish:=True;
      End;
    RightSlope:=(Dx)/(Dy);
    XLeft :=Box[ObjNo].VertexBank;
     [Poly.Vertex[LeftStart]].Xp+LeftSlope;
    XRight:=Box[ObjNo].VertexBank;
     [Poly.Vertex[RightStart]].Xp+RightSlope;
    While (Not Finish) do
      Begin
        XL:=Round(XLeft);
        XR:=Round(XRight);

        If XL>XR then Finish:=True;
        If XL<0 then XL:=0;
        If XR>319 then XR:=319;
        If (YCtr>=0) and (Not Finish) then
         If XL=XR then PutPixel(Buff^,XL,YCtr,Poly.Color) else
          If XL<XR then
            HLines(Buff^,XL,XR,YCtr,Poly.Color);
        XLeft:=XLeft+LeftSlope;
        XRight:=XRight+RightSlope;
        Inc(YCtr);
        If YCtr>199 then Finish:=True;
        If Not Finish then
```

```
            If (YCtr>Box[ObjNo].VertexBank;
               [Poly.Vertex[LeftEnd]].Yp) then
             Begin
               Repeat
                 LeftStart:=LeftStart+1;
                 If LeftStart>=Poly.TotalVertex then LeftStart:=0;
                 LeftEnd:=LeftStart+1;
                 If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
                 Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                    Xp-Box[ObjNo].VertexBank;
                      [Poly.Vertex[LeftStart]].Xp;
                  Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                    Yp-Box[ObjNo].VertexBank;
                      [Poly.Vertex[LeftStart]].Yp+1;
                 If LeftStart=YMaxIndex then Finish:=True;
               Until (Dy<>0) or Finish;
               If Not Finish then
                 Begin
                   LeftSlope:=(Dx)/(Dy);
                   XLeft:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[LeftStart]].Xp+LeftSlope;
                 End;
             End;
           If Not Finish then
             Begin
               If YCtr>Box[ObjNo].VertexBank;
                 [Poly.Vertex[RightEnd]].Yp then
                 Begin
                   Repeat
                     RightStart:=RightStart-1;
                     If RightStart<0 then;
                      RightStart:=Poly.TotalVertex-1;
                     RightEnd:=RightStart-1;
                     If RightEnd<0 then;
                      RightEnd:=Poly.TotalVertex-1;
                     Dx:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[RightEnd]].;
                        Xp-Box[ObjNo].VertexBank;
                           [Poly.Vertex[RightStart]].Xp;
                     Dy:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[RightEnd]].;
                        Yp-Box[ObjNo].VertexBank;
                           [Poly.Vertex[RightStart]].Yp+1;
                     If RightStart=YMaxIndex then Finish:=True;
                   Until (Dy<>0) or Finish;
                   If Not Finish then
                     Begin
                       RightSlope:=(Dx)/(Dy);
                       XRight:=Box[ObjNo].VertexBank;
                          [Poly.Vertex[RightStart]].Xp;
                       XRight:=XRight+RightSlope;
                     End;
                 End;
             End;
         End;
     End;
  End;

Function Vertical_VV_Testing(YMin,YMax:Real) : Boolean;
  Begin
    Vertical_VV_Testing:=False;
```

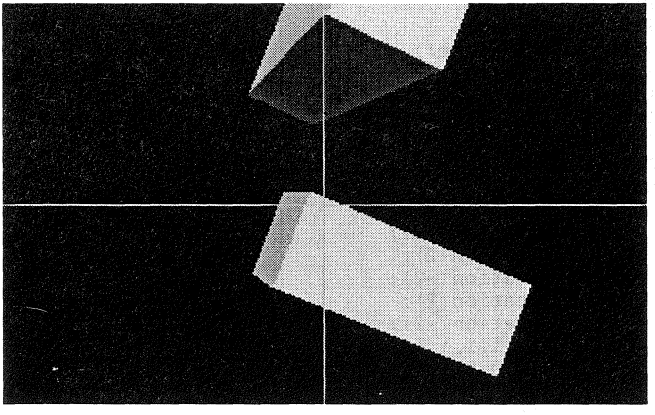
```
    If (YMin>199) or (YMax<0) then Vertical_VV_Testing:=True;
  End;

Function Horisontal_VV_Testing(XMin,XMax:Real) : Boolean;
  Begin
    Horisontal_VV_Testing:=False;
    If (XMin>319) or (XMax<0) then Horisontal_VV_Testing:=True;
  End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        Poly_Shading(Box[ObjNo].Polygon[PolyNo],ObjNo);
      End;
  End;

Procedure View_Volume_Filter;
Var A,B:Integer;
  Begin
    For A:=0 to ListCtr-1 do
      If PolygonList[A].Active then
        Begin
          If PolygonList[A].ZMax<=100 then;
             PolygonList[A].Active:=False;
          If PolygonList[A].ZMin<=50  then;
             PolygonList[A].Active:=False;
          If Vertical_VV_Testing(PolygonList[A].YMin,;
             PolygonList[A].YMax)
            then PolygonList[A].Active:=False;
          If Horisontal_VV_Testing(PolygonList[A].XMin,;
             PolygonList[A].XMax)
            then PolygonList[A].Active:=False;
        End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
```

```
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        Box[A].X:=Box[A].X+Tx;
        Box[A].Y:=Box[A].Y+Ty;
        Box[A].Z:=Box[A].Z+Tz;
      End;
  End;

Procedure Painter_Sort;
```

```
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
    For A:=0 to ListCtr-2 do
     If PolygonList[A].active then
      For B:=A+1 to ListCtr-1 do
        If PolygonList[B].active then
          If PolygonList[A].ZMin<PolygonList[B].ZMin then
            Begin
              Bf:=PolygonList[A];
              PolygonList[A]:=PolygonList[B];
              PolygonList[B]:=Bf;
            End;
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Ch:='s';
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Object;
        View_Volume_Filter;
        Painter_Sort;
        Display_PList_Shading;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          'r':Scale(1.1,1,1);
          'f':Scale(0.9,1,1);
          't':Scale(1,1.1,1);
          'g':Scale(1,0.9,1);
          'y':Scale(1,1,1.1);
          'h':Scale(1,1,0.9);
          '7':Dilatasi(4,0,0);
          '1':Dilatasi(-4,0,0);
          '8':Dilatasi(0,4,0);
          '2':Dilatasi(0,-4,0);
          '9':Dilatasi(0,0,4);
          '3':Dilatasi(0,0,-4);
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```



Pada saat ini ketiga objek masih berada pada ruang penglihatan. Ketiganya masih digambar.



Pada saat ini, objek terdekat telah berada pada koordinat Z negatif. Yang berarti menembus ruang batas Z. Sehingga poligon dari objek tersebut tidak digambar karena telah dicoret dari daftar polygon list.

PENJELASAN LISTING PROGRAM

Yuk kita bahas modifikasi dari program di atas.

Pertama kali dilakukan sedikit perubahan pada data structure dari polygon list.

```
PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
```

Ditambahkan XMin, XMax, YMin dan YMax untuk menyimpan koordinat X

dan Y dua dimensi yang terbesar dan terkecil dari sebuah poligon di dalam polygon list. Variabel-variabel inilah yang memegang peranan penting dalam melakukan View Volume Filter.

```
    ZMin        : Real;
    ZMax        : Real;
    XMin        : Real;
    XMax        : Real;
    YMin        : Real;
    YMax        : Real;
    Color       : Integer;
  End;
```

Modifikasi kedua ialah dengan menambahkan dua buah function di bawah ini.

Function ini berguna untuk mendeteksi apakah sebuah poligon dengan YMin dan YMax yang dikirimkan lewat parameter berada di luar atau di dalam bidang pembatasan Y.

```
Function Vertical_VV_Testing(YMin,YMax:Real) : Boolean;
  Begin
    Vertical_VV_Testing:=False;
```

Jika titik yang berkoordinat Y terkecil dari poligon berada di bawah layar maka seluruh bagian poligon tersebut pasti di bawah layar. Demikian juga jika titik Y terbesar berada di atas layar, maka seluruh bagian poligon berada di atas layar.

```
    If (YMin>199) or (YMax<0) then Vertical_VV_Testing:=True;
  End;

Function Horizontal_VV_Testing(XMin,XMax:Real): Boolean;
  Begin
    Horizontal_VV_Testing:=False;
```

Jika titik yang berkoordinat X terkecil dari poligon berada di kanan layar maka seluruh bagian poligon tersebut pasti di kanan layar. Demikian juga jika titik X terbesar berada di kiri layar, maka seluruh bagian poligon berada di kiri layar.

```
    If (XMin>319) or (XMax<0) then Horizontal_VV_Testing:=True;
  End;
```

Modifikasi terakhir ialah dengan menambahkan procedure View_Volume_Filter.

```
Procedure View_Volume_Filter;
Var A,B:Integer;
  Begin
```

Lakukan scanning terhadap seluruh poligon di dalam polygon list.

```
For A:=0 to ListCtr-1 do
  If PolygonList[A].Active then
    Begin
```

Untuk menghindari pembagian dengan 0 (division by 0) maka kita tidak menggunakan koordinat 0 sebagai pembatas Z melainkan mengambil koordinat Z = 100 sebagai pembatas.

```
If PolygonList[A].ZMax<=100 then;
   PolygonList[A].Active:=False;
```

Pembatasan ini hanya untuk berjaga-jaga, jangan sampai ada poligon yang melewati batas koordinat Z = 50. Tujuannya ialah menghindari poligon yang separuh berada di luar batas layar dan separuh di dalam batas layar. Hanya untuk pembatasan Z, perlu dilakukan hal ini karena hal ini sering mengakibatkan program error karena terjadinya pembagian dengan 0.

```
If PolygonList[A].ZMin<=50  then;
   PolygonList[A].Active:=False;
```

Lakukan test terhadap pembatasan Y.

```
If Vertical_VV_Testing;
  (PolygonList[A].YMin,PolygonList[A].YMax)
  then PolygonList[A].Active:=False;
```

Lakukan test terhadap pembatasan X.

```
If Horizontal_VV_Testing;
  (PolygonList[A].XMin,PolygonList[A].XMax)
  then PolygonList[A].Active:=False;
End;
End;
```

Eh, ada satu modifikasi lagi ... yaitu memanggil procedure View_Volume_Filter di main program.

Yak, sampai di sinilah pembahasan kita mengenai filter. Sebenarnya masih ada sebuah filter lagi yang disebut Volume Clipping. Tujuan dari filter ini ialah memotong poligon yang memotong batas layar. Dengan ini poligon yang berada di antara batas Z dengan algorithm kita kali ini yang kita coret dari daftar polygon list, akan dipotong menjadi poligon baru yang seluruhnya berada di dalam ruang penglihatan. Teknik ini cukup rumit dan tidak akan dibahas pada buku ini, karena sebenarnya teknik ini jarang sekali dipergunakan oleh para pemrogram game mengingat kalkulasi yang dilakukan cukup panjang, sehingga memperlambat jalannya animasi. Dalam hal ini biasanya volume clipping ditiadakan.

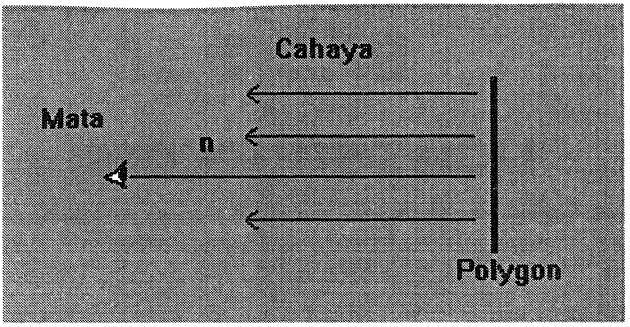
BAB 7
LIGHT SOURCING

7.1 Pendahuluan

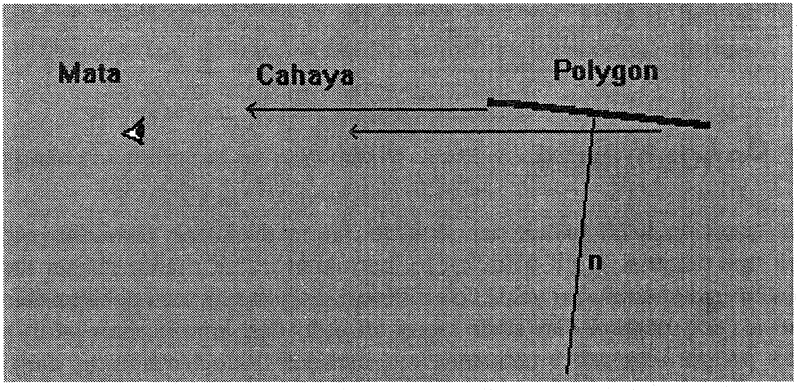
Light sourcing ialah teknik pencahayaan yang dilakukan terhadap sebuah objek tiga dimensi. Light sourcing ini bertujuan untuk memperbesar tingkat realistisitas dari suatu proyeksi tiga dimensi. Logikanya ialah sebagai berikut. Cahaya yang berasal dari sinar matahari atau dari lampu akan jatuh pada setiap poligon dengan intensitas yang berbeda-beda tergantung dari arah hadap dari poligon, juga tergantung dari jarak antara poligon dengan sumber cahaya.

Ada beberapa macam light sourcing. Di antaranya ialah Flat Shading, Goraud Shading dan Pong Shading. Pada pembahasan kita kali ini, akan dijelaskan mengenai light sourcing yang paling sederhana yaitu Flat Shading.

Pada Flat Shading, dihitung intensitas cahaya yang jatuh pada sebuah poligon dengan cara melihat arah hadap (garis normal) dari poligon tersebut. Jika poligon tersebut berhadapan langsung dengan sumber cahaya, dalam hal ini garis normal sejajar dengan garis pandang kita (garis pandang di sini sama dengan garis pandang yang kita bahas pada Bab 4) atau sudut antara garis normal dengan garis pandang ialah 0 maka poligon tersebut mendapat intensitas cahaya maximum. Secara fisika hal ini terjadi karena banyak sinar yang dipantulkan ke mata kita. Untuk poligon yang memiliki sudut antara garis normal dengan garis pandang kita hampir 90 derajat, memiliki intensitas cahaya minimum karena sangat sedikit sinar yang terpantul ke mata kita.



Poligon yang garis normal (n)-nya sejajar dengan garis pandang kita, akan mendapat intensitas cahaya maksimum.



Poligon yang memiliki garis normal (n) hampir tegak lurus dengan garis pandang kita, akan mendapat intensitas cahaya minimum.

Dalam hal ini diasumsikan jenis sumber cahaya ialah ambient yang berarti cahaya lingkungan (berasal dari segala arah). Yah anggaplah cahaya matahari yang telah terpantul-pantul. Sebenarnya dalam melakukan perhitungan secara detil kita harus memperhitungkan pemantulan cahaya secara diffusi, difraksi gelombang, dispersi gelombang, absorption dan lain-lainnya. Tetapi kita khan bukannya profesor yang sedang membuat pesawat ruang angkasa khan ..., kita hanyalah pemrogram game yang membuat game komputer, jadi nggak perlu terlalu detil.

Tentu saja dalam menangani hal ini, kita cukup menggunakan kalkulasi sudut antara garis pandang dengan garis normal tiap poligon.

Yuk kita coba.

```
{-------------------------------------------------}
{ Program     : B3SHADE7.PAS                      }
{ Coder       : Agustinus Nalwan                  }
{ Description: Program transformasi objek         }
{              dengan flat shadded ambient        }
{              light sourcing                     }
{-------------------------------------------------}

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       X, Y, Z : Real;
       Xp,Yp   : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
       X,Y,Z      : Real;
     End;

     PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
       ZMin         : Real;
       ZMax         : Real;
       XMin         : Real;
       XMax         : Real;
       YMin         : Real;
       YMax         : Real;
       Color        : Integer;
     End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ListCtr:Integer;
    buff:Pointer;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
```

```
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].X:=-lx;Box[No].VertexBank[0].Y:=ly;
        Box[No].VertexBank[0].Z:=-lz;
    Box[No].VertexBank[1].X:=lx;Box[No].VertexBank[1].Y:=ly;
        Box[No].VertexBank[1].Z:=-lz;
    Box[No].VertexBank[2].X:=lx;Box[No].VertexBank[2].Y:=-ly;
        Box[No].VertexBank[2].Z:=-lz;
    Box[No].VertexBank[3].X:=-lx;Box[No].VertexBank[3].Y:=-ly;
        Box[No].VertexBank[3].Z:=-lz;
    Box[No].VertexBank[4].X:=-lx;Box[No].VertexBank[4].Y:=ly;
        Box[No].VertexBank[4].Z:=lz;
    Box[No].VertexBank[5].X:=lx;Box[No].VertexBank[5].Y:=ly;
        Box[No].VertexBank[5].Z:=lz;
    Box[No].VertexBank[6].X:=lx;Box[No].VertexBank[6].Y:=-ly;
        Box[No].VertexBank[6].Z:=lz;
    Box[No].VertexBank[7].X:=-lx;Box[No].VertexBank[7].Y:=-ly;
        Box[No].VertexBank[7].Z:=lz;
    Box[No].Polygon[0].TotalVertex:=4;
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;

    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;
```

```
    Box[No].Polygon[0].Color:=17;
    Box[No].Polygon[1].Color:=19;
    Box[No].Polygon[2].Color:=21;
    Box[No].Polygon[3].Color:=23;
    Box[No].Polygon[4].Color:=25;
    Box[No].Polygon[5].Color:=27;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].X:=X;Box[No].Y:=Y;Box[No].Z:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,0,0,550,50,50,50,13);
    Create_Box(1,90,0,850,20,50,30,11);
    Create_Box(2,80,130,850,50,20,100,12);
    Remove_PolyList;
    For A:=1 to 91 do
      Begin
        SetVisualPalette(A,63-Round(A/1.5),63-Round(A/1.5),;
            63-Round(A/1.5));
      End;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
```

```
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
            PolygonList[ListCtr].ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].XMax:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr].XMin:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr].YMax:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            PolygonList[ListCtr].YMin:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr].ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr].ZMin>VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;

                If PolygonList[ListCtr].YMax<VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                    PolygonList[ListCtr].YMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr].YMin>VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                    PolygonList[ListCtr].YMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr].XMax<VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                    PolygonList[ListCtr].XMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Xp;
                If PolygonList[ListCtr].XMin>VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                    PolygonList[ListCtr].XMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Xp;
              End;
            Inc(ListCtr);
          End;
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        With Box[ObjNo] do
          Begin
```

```
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;        Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo,Color:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
    LeftCtr     : Integer;
    XL,XR,Bf    : Longint;
    Dx,Dy       : Real;
    Finish      : Boolean;
    Ty          : Longint;
  Begin
    YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMinIndex:=0;
    YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMaxIndex:=0;

    For A:=1 to Poly.TotalVertex-1 do
      Begin
        Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMinIndex:=A;
          End;
        If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMaxIndex:=A;
          End;
      End;

    LeftStart:=YMinIndex; RightStart:=YMinIndex;
    YCtr :=YMin;
    LeftEnd:=LeftStart+1;
    If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
```

```
RightEnd:=RightStart-1;
If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
   Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
   Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
Finish:=False;
While (Dy=0) and (Not Finish) do
  Begin
    LeftStart:=LeftStart+1;
    If LeftStart>=Poly.TotalVertex then LeftStart:=0;
    LeftEnd:=LeftStart+1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
      Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
      Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
    If LeftStart=YMaxIndex then Finish:=True;
  End;
If Not Finish then
  Begin
    LeftSlope:=(Dx)/(Dy);
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
      Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
      Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
    While (Dy=0) and (Not Finish) do
      Begin
        RightStart:=RightStart-1;
        If RightStart<0 then RightStart:=Poly.TotalVertex-1;
        RightEnd:=RightStart-1;
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
          Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
          Yp-Box[ObjNo].VertexBank;
            [Poly.Vertex[RightStart]].Yp+1;
        If RightStart=YMaxIndex then Finish:=True;
      End;
    RightSlope:=(Dx)/(Dy);
    XLeft :=Box[ObjNo].VertexBank;
        [Poly.Vertex[LeftStart]].Xp+LeftSlope;
    XRight:=Box[ObjNo].VertexBank;
        [Poly.Vertex[RightStart]].Xp+RightSlope;
    While (Not Finish) do
      Begin
        XL:=Round(XLeft);
        XR:=Round(XRight);

        If XL>XR then Finish:=True;
        If XL<0 then XL:=0;
        If XR>319 then XR:=319;
        If (YCtr>=0) and (Not Finish) then
         If XL=XR then PutPixel(Buff^,XL,YCtr,Color) else
          If XL<XR then
            HLines(Buff^,XL,XR,YCtr,Color);
        XLeft:=XLeft+LeftSlope;
        XRight:=XRight+RightSlope;
        Inc(YCtr);
        If YCtr>199 then Finish:=True;
        If Not Finish then
         If (YCtr>Box[ObjNo].VertexBank;
           [Poly.Vertex[LeftEnd]].Yp) then
```

```
            Begin
              Repeat
                LeftStart:=LeftStart+1;
                If LeftStart>=Poly.TotalVertex then LeftStart:=0;
                LeftEnd:=LeftStart+1;
                If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
                Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                  Xp-Box[ObjNo].VertexBank;
                    [Poly.Vertex[LeftStart]].Xp;
                Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
                  Yp-Box[ObjNo].VertexBank;
                    [Poly.Vertex[LeftStart]].Yp+1;
                If LeftStart=YMaxIndex then Finish:=True;
              Until (Dy<>0) or Finish;
              If Not Finish then
                Begin
                  LeftSlope:=(Dx)/(Dy);
                  XLeft:=Box[ObjNo].VertexBank;
                    [Poly.Vertex[LeftStart]].Xp+LeftSlope;
                End;
            End;
          If Not Finish then
            Begin
              If YCtr>Box[ObjNo].VertexBank;
                [Poly.Vertex[RightEnd]].Yp then
                Begin
                  Repeat
                    RightStart:=RightStart-1;
                    If RightStart<0 then;
                      RightStart:=Poly.TotalVertex-1;
                    RightEnd:=RightStart-1;
                    If RightEnd<0 then;
                      RightEnd:=Poly.TotalVertex-1;
                    Dx:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[RightEnd]].;
                        Xp-Box[ObjNo].VertexBank;
                          [Poly.Vertex[RightStart]].Xp;
                    Dy:=Box[ObjNo].VertexBank;
                      [Poly.Vertex[RightEnd]].;
                        Yp-Box[ObjNo].VertexBank;
                          [Poly.Vertex[RightStart]].Yp+1;
                    If RightStart=YMaxIndex then Finish:=True;
                  Until (Dy<>0) or Finish;
                  If Not Finish then
                    Begin
                      RightSlope:=(Dx)/(Dy);
                      XRight:=Box[ObjNo].VertexBank;
                        [Poly.Vertex[RightStart]].Xp;
                      XRight:=XRight+RightSlope;
                    End;
                End;
            End;
        End;
    End;
  End;

Function Vertical_VV_Testing(YMin,YMax:Real) : Boolean;
  Begin
    Vertical_VV_Testing:=False;
    If (YMin>199) or (YMax<0) then Vertical_VV_Testing:=True;
  End;
```

```
Function Horisontal_VV_Testing(XMin,XMax:Real) : Boolean;
  Begin
    Horisontal_VV_Testing:=False;
    If (XMin>319) or (XMax<0) then Horisontal_VV_Testing:=True;
  End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        Poly_Shading(Box[ObjNo].Polygon;
           [PolyNo],ObjNo,PolygonList[A].Color);
      End;
  End;

Procedure View_Volume_Filter;
Var A,B:Integer;
  Begin
    For A:=0 to ListCtr-1 do
      If PolygonList[A].Active then
        Begin
          If PolygonList[A].ZMax<=100 then;
             PolygonList[A].Active:=False;
          If PolygonList[A].ZMin<=50  then;
             PolygonList[A].Active:=False;
          If Vertical_VV_Testing;
            (PolygonList[A].YMin,PolygonList[A].YMax)
            then PolygonList[A].Active:=False;
          If Horisontal_VV_Testing;
            (PolygonList[A].XMin,PolygonList[A].XMax)
            then PolygonList[A].Active:=False;
        End;
  End;

Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
    Deg             :Real;
    Intensity       :Integer;
  Begin
    View[0]:=0;View[1]:=0;View[2]:=10;
    For Ctr:=0 to ListCtr-1 do
      Begin
        ObjNo :=PolygonList[Ctr].ObjNo;
        PolyNo:=PolygonList[Ctr].PolyNo;
        A[0]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[0]].X-
               Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        A[1]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Y-
               Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        A[2]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Z-
```

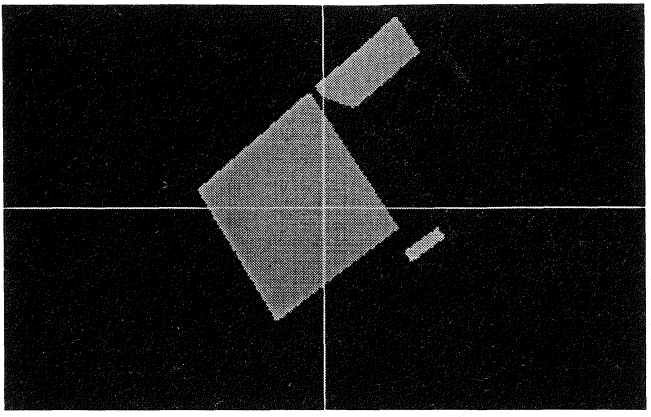
```
              Box[ObjNo].VertexBank;
               [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;

        View[0]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        View[1]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        View[2]:=-Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z-400;

        B[0]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[2]].X-
               Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        B[1]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Y-
               Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        B[2]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Z-
               Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;
        Vect_Cross(Normal,A,B);
        Deg:=Abs(Abs(Vect_Angle(Normal,View)));
        Intensity:=Round(Deg);
        PolygonList[Ctr].Color:=Intensity+1;
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].X;
            CMat[1]:=VertexBank[d].Y;
            CMat[2]:=VertexBank[d].Z;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].X:=CMat[0];
            VertexBank[d].Y:=CMat[1];
            VertexBank[d].Z:=CMat[2];
          End;
      End;
  End;
```

```
Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        Box[A].X:=Box[A].X+Tx;
        Box[A].Y:=Box[A].Y+Ty;
        Box[A].Z:=Box[A].Z+Tz;
      End;
  End;

Procedure Painter_Sort;
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
    For A:=0 to ListCtr-2 do
     If PolygonList[A].active then
      For B:=A+1 to ListCtr-1 do
        If PolygonList[B].active then
          If PolygonList[A].ZMin<PolygonList[B].ZMin then
            Begin
              Bf:=PolygonList[A];
              PolygonList[A]:=PolygonList[B];
```

```
            PolygonList[B]:=Bf;
          End;
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Ch:='s';
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        Project_Object;
        BackFace_Filter;
        View_Volume_Filter;
        Painter_Sort;
        Display_PList_Shading;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          'r':Scale(1.1,1,1);
          'f':Scale(0.9,1,1);
          't':Scale(1,1.1,1);
          'g':Scale(1,0.9,1);
          'y':Scale(1,1,1.1);
          'h':Scale(1,1,0.9);
          '7':Dilatasi(4,0,0);
          '1':Dilatasi(-4,0,0);
          '8':Dilatasi(0,4,0);
          '2':Dilatasi(0,-4,0);
          '9':Dilatasi(0,0,4);
          '3':Dilatasi(0,0,-4);
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Cobalah gerakkan objek pada dunia tiga dimensi. Di sini terlihat bahwa poligon yang tegak lurus terhadap kita akan mendapat cahaya maksimum. Dan semakin kita memutarnya menjauhi tegak lurus, poligon tersebut akan meredup.



Perhatikan, poligon yang menghadap ke kita jauh lebih terang dari poligon yang lain.

PENJELASAN LISTING PROGRAM

Tidak banyak yang perlu diubah dalam menangani masalah flat shading ini.

Pertama lihatlah procedure Initialize. Di sana dilakukan pengaturan palette mulai nomor 1 hingga 91 yang diisi dengan warna mulai putih terang hingga hitam. Tujuannya ialah menghasilkan 90 intensitas warna putih untuk mewakili setiap satu derajat sudut antara garis normal dengan garis pandang.

```
For A:=1 to 91 do
      Begin
        SetVisualPalette(A,63-Round(A/1.5),63-Round(A/1.5),63-
Round(A/1.5));
      End;
  End;
```

Pembenahan kedua ialah dengan memanfaatkan procedure BackFace_Filter kita yang lama untuk menghitung intensitas cahaya dari sebuah poligon.

```
Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
    Deg             :Real;
    Intensity       :Integer;
  Begin
    View[0]:=0;View[1]:=0;View[2]:=10;
```

Lakukan scanning terhadap seluruh poligon dalam polygon list.

```
For Ctr:=0 to ListCtr-1 do
  Begin
    ObjNo :=PolygonList[Ctr].ObjNo;
    PolyNo:=PolygonList[Ctr].PolyNo;
    A[0]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[0]].X-
          Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
    A[1]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Y-
          Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
    A[2]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Z-
          Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;

    View[0]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
    View[1]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
    View[2]:=-Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z-400;

    B[0]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[2]].X-
          Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
    B[1]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Y-
          Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
    B[2]:=Box[ObjNo].VertexBank;
     [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Z-
          Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;
```

Dengan menggunakan dua buah garis pada poligon, cari garis normal.

```
Vect_Cross(Normal,A,B);
```

Hitung sudut antara garis normal dengan garis pandang.

```
Deg:=Abs(Abs(Vect_Angle(Normal,View)));
```

Sudut yang dihasilkan yang berkisar antara 0 hingga 90 derajat ialah intensitas cahaya atau nomor warna untuk poligon tersebut. Ingat nomor warna mulai 1 hingga 91 telah diatur, sehingga warna 1 ialah putih paling terang sedangkan warna 91 ialah hitam.

```
Intensity:=Round(Deg);
```

Tambahkan intensitas yang ditemukan dengan 1 karena warna yang disediakan berkisar antara 1 hingga 91.

```
      PolygonList[Ctr].Color:=Intensity+1;
    End;
End;
```

BAB 8

3D INVERSE TRANSFORMATION

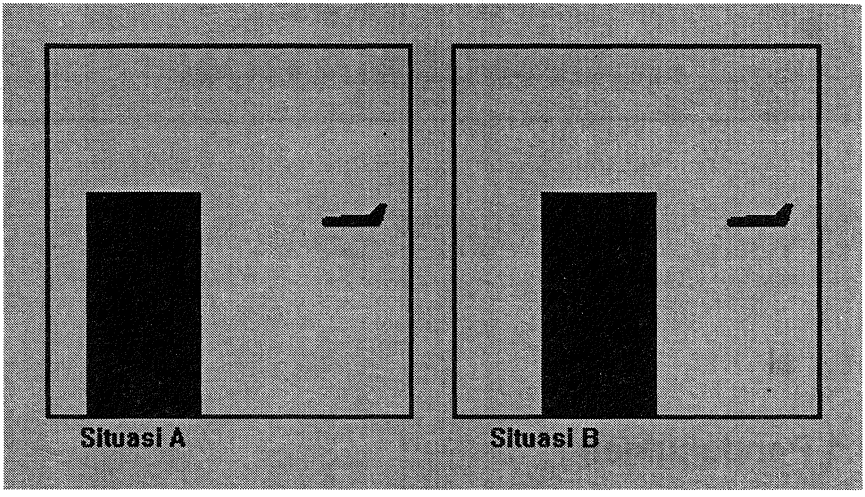
8.1 Pendahuluan

Sejauh ini kita telah mempelajari bagaimana cara memproyeksikan, mentransformasikan sebuah objek dengan beberapa metode. Tetapi semua ini belumlah cukup untuk merakit sebuah game flight simulator seperti yang telah dijual di pasaran. Ingat program terakhir yang kita buat yaitu B3SHADE7 di mana kita mampu mengontrol transformasi dari seluruh objek sementara kita memonitor dari sebuah koordinat? Pada game flight simulator yang sebenarnya, tidak hanya objek pada dunia tiga dimensi saja yang dapat bergerak melainkan kita sebagai penglihat juga dapat bergerak. Dan seiring dengan gerakan kita, sudut pandang kita juga ikut bergerak. Hmmm fungsi ini belum pernah kita pelajari.

Untuk mempermudah pembahasan teknik ini, bayangkan situasi pada saat kita naik mobil. Kadang pada saat ngantuk, mobil kita dalam keadaan diam dan mobil sebelah kita bergerak perlahan ke depan. Secara tiba-tiba terlintas pada pikiran kita bahwa mobil kita bergerak mundur perlahan. Efek ini terjadi karena teori relativitas. Teori relativitas ditemukan oleh Einstein seorang ilmuwan fisika yang sangat cerdas. Beliau lahir di Jerman. Semenjak masa kecilnya Einstein dianggap sebagai orang yang bodoh karena ... eh maaf. Kembali ke pokok permasalahan. Sampai mana kita tadi?

Oh ya secara tiba-tiba terlintas pikiran bahwa mobil kita bergerak mundur. Pada saat itu, otak kita masih belum dapat membedakan mana yang bergerak ... Mobil kita atau mobil sebelah.

Nah, dengan memanfaatkan efek ini ... pada saat kita sebagai pelihat pada dunia tiga dimensi ... jika kita maju mendekati sebuah objek, maka dengan teori relativitas kita dapat menggeser objek tersebut mendekati kita bukannya kita yang mendekatinya.



Selain melakukan penggeseran lokasi pesawat, kita dapat melakukan penggeseran lokasi gedung untuk menimbulkan efek bahwa pesawat mendekati gedung.

Demikian pula dengan rotasi. Jika kita melakukan rotasi terhadap sumbu X misalnya, maka kita dapat merotasikan seluruh objek terhadap sumbu X dengan arah berlawanan.

Mudah kan?

Untuk itu perlu dilakukan modifikasi pada proses transformasi dari objek. Sebelum objek diproyeksikan, dilakukan inverse transformasi dari posisi dan sudut pandang si pelihat pada saat itu. Dengan ini berarti kita telah menggerakkan seluruh objek di dunia tiga dimensi relative terhadap diri kita.

Untuk lebih jelasnya perhatikan program berikut.

```
{-----------------------------------------------}
{ Program     : B3VIEW.PAS                      }
{ Coder       : Agustinus Nalwan                }
{ Description: Program proyeksi dengan menggunakan }
{              sudut pandang camera.            }
{-----------------------------------------------}

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..10] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       XR, YR, ZR : Real;
       X, Y, Z    : Real;
       Xp,Yp      : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
       X,Y,Z      : Real;
       XR,YR,ZR   : Real;
     End;

     PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
       ZMin         : Real;
       ZMax         : Real;
       XMin         : Real;
       XMax         : Real;
       YMin         : Real;
       YMax         : Real;
       Color        : Integer;
     End;

     ViewPort_Rec = Record
       X, Y, Z  : Real;
       Dx,Dy,Dz : Real;
     End;

Var A,B:Integer;
    Box:array[0..2] of ObjRect;
    PolygonList:array[0..100] of PolyList_Rec;
    ViewPort:ViewPort_Rec;
    ListCtr:Integer;
    buff:Pointer;
```

```
Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].XR:=-lx;Box[No].VertexBank[0].YR:=ly;
        Box[No].VertexBank[0].ZR:=-lz;
    Box[No].VertexBank[1].XR:=lx;Box[No].VertexBank[1].YR:=ly;
        Box[No].VertexBank[1].ZR:=-lz;
    Box[No].VertexBank[2].XR:=lx;Box[No].VertexBank[2].YR:=-ly;
        Box[No].VertexBank[2].ZR:=-lz;
    Box[No].VertexBank[3].XR:=-lx;Box[No].VertexBank[3].YR:=-ly;
        Box[No].VertexBank[3].ZR:=-lz;
    Box[No].VertexBank[4].XR:=-lx;Box[No].VertexBank[4].YR:=ly;
        Box[No].VertexBank[4].ZR:=lz;
    Box[No].VertexBank[5].XR:=lx;Box[No].VertexBank[5].YR:=ly;
        Box[No].VertexBank[5].ZR:=lz;
    Box[No].VertexBank[6].XR:=lx;Box[No].VertexBank[6].YR:=-ly;
        Box[No].VertexBank[6].ZR:=lz;
    Box[No].VertexBank[7].XR:=-lx;Box[No].VertexBank[7].YR:=-ly;
        Box[No].VertexBank[7].ZR:=lz;
    Box[No].Polygon[0].TotalVertex:=4;
    Box[No].Polygon[1].TotalVertex:=4;
    Box[No].Polygon[2].TotalVertex:=4;
    Box[No].Polygon[3].TotalVertex:=4;
    Box[No].Polygon[4].TotalVertex:=4;
    Box[No].Polygon[5].TotalVertex:=4;

    Box[No].Polygon[0].Vertex[0]:=0;
    Box[No].Polygon[0].Vertex[1]:=3;
    Box[No].Polygon[0].Vertex[2]:=2;
    Box[No].Polygon[0].Vertex[3]:=1;

    Box[No].Polygon[1].Vertex[0]:=0;
    Box[No].Polygon[1].Vertex[1]:=1;
    Box[No].Polygon[1].Vertex[2]:=5;
    Box[No].Polygon[1].Vertex[3]:=4;

    Box[No].Polygon[2].Vertex[0]:=4;
    Box[No].Polygon[2].Vertex[1]:=5;
    Box[No].Polygon[2].Vertex[2]:=6;
    Box[No].Polygon[2].Vertex[3]:=7;

    Box[No].Polygon[3].Vertex[0]:=7;
    Box[No].Polygon[3].Vertex[1]:=6;
    Box[No].Polygon[3].Vertex[2]:=2;
    Box[No].Polygon[3].Vertex[3]:=3;

    Box[No].Polygon[4].Vertex[0]:=4;
    Box[No].Polygon[4].Vertex[1]:=7;
    Box[No].Polygon[4].Vertex[2]:=3;
```

```
    Box[No].Polygon[4].Vertex[3]:=0;

    Box[No].Polygon[5].Vertex[0]:=5;
    Box[No].Polygon[5].Vertex[1]:=1;
    Box[No].Polygon[5].Vertex[2]:=2;
    Box[No].Polygon[5].Vertex[3]:=6;

    Box[No].Polygon[0].Color:=17;
    Box[No].Polygon[1].Color:=19;
    Box[No].Polygon[2].Color:=21;
    Box[No].Polygon[3].Color:=23;
    Box[No].Polygon[4].Color:=25;
    Box[No].Polygon[5].Color:=27;

    Box[No].TotalPoly:=6;
    Box[No].TotalVertex:=8;
    Box[No].XR:=X;Box[No].YR:=Y;Box[No].ZR:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A].Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Create_Box(0,30,40,550,50,50,50,13);
    Create_Box(1,190,0,850,20,50,30,11);
    Create_Box(2,-100,130,850,50,20,100,12);
    Remove_PolyList;
    For A:=1 to 91 do
      Begin
        SetVisualPalette(A,63-Round(A/1.5),63-Round(A/1.5),;
            63-Round(A/1.5));
      End;
    With ViewPort do
      Begin
        X:=0;Y:=0;Z:=0;Dx:=0;Dy:=0;Dz:=0;
      End;
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
```

```
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr].ObjNo:=No;
            PolygonList[ListCtr].PolyNo:=A;
            PolygonList[ListCtr].Active:=True;
            PolygonList[ListCtr].ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr].XMax:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr].XMin:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr].YMax:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            PolygonList[ListCtr].YMin:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr].ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr].ZMin>VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                    PolygonList[ListCtr].ZMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;

                If PolygonList[ListCtr].YMax<VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                    PolygonList[ListCtr].YMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr].YMin>VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                    PolygonList[ListCtr].YMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr].XMax<VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                    PolygonList[ListCtr].XMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Xp;
                If PolygonList[ListCtr].XMin>VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                    PolygonList[ListCtr].XMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Xp;
              End;
            Inc(ListCtr);
          End;
      End;
```

```
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;        Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo,Color:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
    LeftCtr     : Integer;
    XL,XR,Bf    : Longint;
    Dx,Dy       : Real;
    Finish      : Boolean;
    Ty          : Longint;
  Begin
    YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMinIndex:=0;
    YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMaxIndex:=0;

    For A:=1 to Poly.TotalVertex-1 do
      Begin
        Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
            YMinIndex:=A;
```

```
      End;
    If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
      Begin
        YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        YMaxIndex:=A;
      End;
  End;

LeftStart:=YMinIndex; RightStart:=YMinIndex;
YCtr :=YMin;
LeftEnd:=LeftStart+1;
If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
RightEnd:=RightStart-1;
If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
   Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
   Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
Finish:=False;
While (Dy=0) and (Not Finish) do
  Begin
    LeftStart:=LeftStart+1;
    If LeftStart>=Poly.TotalVertex then LeftStart:=0;
    LeftEnd:=LeftStart+1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
      Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
      Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
    If LeftStart=YMaxIndex then Finish:=True;
  End;
If Not Finish then
  Begin
    LeftSlope:=(Dx)/(Dy);
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
      Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
      Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
    While (Dy=0) and (Not Finish) do
      Begin
        RightStart:=RightStart-1;
        If RightStart<0 then RightStart:=Poly.TotalVertex-1;
        RightEnd:=RightStart-1;
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
           Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
           Yp-Box[ObjNo].VertexBank;
            [Poly.Vertex[RightStart]].Yp+1;
        If RightStart=YMaxIndex then Finish:=True;
      End;
    RightSlope:=(Dx)/(Dy);
    XLeft :=Box[ObjNo].VertexBank;
       [Poly.Vertex[LeftStart]].Xp+LeftSlope;
    XRight:=Box[ObjNo].VertexBank;
       [Poly.Vertex[RightStart]].Xp+RightSlope;
    While (Not Finish) do
      Begin
        XL:=Round(XLeft);
        XR:=Round(XRight);

        If XL>XR then Finish:=True;
```

```
If XL<0 then XL:=0;
If XR>319 then XR:=319;
If (YCtr>=0) and (Not Finish) then
 If XL=XR then PutPixel(Buff^,XL,YCtr,Color) else
  If XL<XR then
    HLines(Buff^,XL,XR,YCtr,Color);
XLeft:=XLeft+LeftSlope;
XRight:=XRight+RightSlope;
Inc(YCtr);
If YCtr>199 then Finish:=True;
If Not Finish then
 If (YCtr>Box[ObjNo].VertexBank;
   [Poly.Vertex[LeftEnd]].Yp) then
  Begin
    Repeat
      LeftStart:=LeftStart+1;
      If LeftStart>=Poly.TotalVertex then LeftStart:=0;
      LeftEnd:=LeftStart+1;
      If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
      Dx:=Box[ObjNo].VertexBank;
         [Poly.Vertex[LeftEnd]].;
          Xp-Box[ObjNo].VertexBank;
           [Poly.Vertex[LeftStart]].Xp;
      Dy:=Box[ObjNo].VertexBank;
         [Poly.Vertex[LeftEnd]].;
          Yp-Box[ObjNo].VertexBank;
           [Poly.Vertex[LeftStart]].Yp+1;
      If LeftStart=YMaxIndex then Finish:=True;
    Until (Dy<>0) or Finish;
    If Not Finish then
      Begin
        LeftSlope:=(Dx)/(Dy);
        XLeft:=Box[ObjNo].VertexBank;
           [Poly.Vertex[LeftStart]].Xp+LeftSlope;
      End;
  End;
If Not Finish then
  Begin
    If YCtr>Box[ObjNo].VertexBank;
      [Poly.Vertex[RightEnd]].Yp then
      Begin
        Repeat
          RightStart:=RightStart-1;
          If RightStart<0 then;
            RightStart:=Poly.TotalVertex-1;
          RightEnd:=RightStart-1;
          If RightEnd<0 then;
            RightEnd:=Poly.TotalVertex-1;
          Dx:=Box[ObjNo].VertexBank;
            [Poly.Vertex[RightEnd]].;
               Xp-Box[ObjNo].VertexBank;
                   [Poly.Vertex[RightStart]].Xp;
          Dy:=Box[ObjNo].VertexBank;
            [Poly.Vertex[RightEnd]].;
               Yp-Box[ObjNo].VertexBank;
                   [Poly.Vertex[RightStart]].Yp+1;
          If RightStart=YMaxIndex then Finish:=True;
        Until (Dy<>0) or Finish;
        If Not Finish then
          Begin
            RightSlope:=(Dx)/(Dy);
```

```
                        XRight:=Box[ObjNo].VertexBank;
                             [Poly.Vertex[RightStart]].Xp;
                        XRight:=XRight+RightSlope;
                      End;
                  End;
              End;
          End;
      End;
  End;

Function Vertical_VV_Testing(YMin,YMax:Real) : Boolean;
  Begin
    Vertical_VV_Testing:=False;
    If (YMin>199) or (YMax<0) then Vertical_VV_Testing:=True;
  End;

Function Horisontal_VV_Testing(XMin,XMax:Real) : Boolean;
  Begin
    Horisontal_VV_Testing:=False;
    If (XMin>319) or (XMax<0) then Horisontal_VV_Testing:=True;
  End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    ColCtr:=17;
    For A:=0 to ListCtr-1 do
     If PolygonList[A].Active then
      Begin
        ObjNo :=PolygonList[A].ObjNo;
        PolyNo:=PolygonList[A].PolyNo;
        Poly_Shading(Box[ObjNo].Polygon;
           [PolyNo],ObjNo,PolygonList[A].Color);
      End;
  End;

Procedure View_Volume_Filter;
Var A,B:Integer;
  Begin
    For A:=0 to ListCtr-1 do
      If PolygonList[A].Active then
        Begin
          If PolygonList[A].ZMax<=100 then;
             PolygonList[A].Active:=False;
          If PolygonList[A].ZMin<=50  then;
             PolygonList[A].Active:=False;
          If Vertical_VV_Testing;
               (PolygonList[A].YMin,PolygonList[A].YMax)
            then PolygonList[A].Active:=False;
          If Horisontal_VV_Testing;
               (PolygonList[A].XMin,PolygonList[A].XMax)
            then PolygonList[A].Active:=False;
        End;
  End;

Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
    Deg             :Real;
```

```
    Intensity       :Integer;
  Begin
    View[0]:=0;View[1]:=0;View[2]:=10;
    For Ctr:=0 to ListCtr-1 do
      Begin
        ObjNo :=PolygonList[Ctr].ObjNo;
        PolyNo:=PolygonList[Ctr].PolyNo;
        A[0]:=Box[ObjNo].VertexBank[Box[ObjNo].Polygon;
            [PolyNo].Vertex[0]].X-
              Box[ObjNo].VertexBank[Box[ObjNo].Polygon;
                [PolyNo].Vertex[1]].X;
        A[1]:=Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Y-
              Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        A[2]:=Box[ObjNo].VertexBank[Box[ObjNo].Polygon;
            [PolyNo].Vertex[0]].Z-
              Box[ObjNo].VertexBank[Box[ObjNo].Polygon;
                [PolyNo].Vertex[1]].Z;

        View[0]:=Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        View[1]:=Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        View[2]:=-Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z-400;

        B[0]:=Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[2]].X-
              Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        B[1]:=Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Y-
              Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        B[2]:=Box[ObjNo].VertexBank;
            [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Z-
              Box[ObjNo].VertexBank;
                [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;
        Vect_Cross(Normal,A,B);
        Deg:=Abs(Abs(Vect_Angle(Normal,View)));
        Intensity:=Round(Deg);
        PolygonList[Ctr].Color:=Intensity+1;
      End;
  End;

Procedure Display_Try;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
        ObjNo :=0;
        PolyNo:=0;
        Box[ObjNo].VertexBank[0].Xp:=160;
        Box[ObjNo].VertexBank[0].Yp:=30;
        Box[ObjNo].VertexBank[1].Xp:=220;
        Box[ObjNo].VertexBank[1].Yp:=100;
        Box[ObjNo].VertexBank[2].Xp:=160;
        Box[ObjNo].VertexBank[2].Yp:=130;
        Box[ObjNo].VertexBank[3].Xp:=60;
        Box[ObjNo].VertexBank[3].Yp:=100;
        Box[ObjNo].Polygon[0].Vertex[0]:=0;
```

```
        Box[ObjNo].Polygon[0].Vertex[1]:=3;
        Box[ObjNo].Polygon[0].Vertex[2]:=2;
        Box[ObjNo].Polygon[0].Vertex[3]:=1;
        Box[ObjNo].Polygon[0].TotalVertex:=4;
        Box[ObjNo].TotalPoly:=1;
        Box[ObjNo].TotalVertex:=4;
        Poly_Shading(Box[0].Polygon[0],0,15);
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure ViewPort_Transform;
Var A,B:Integer;
    CMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    Mat_XRotate(XMat,-ViewPort.Dx);
    Mat_YRotate(YMat,-ViewPort.Dy);
    Mat_ZRotate(ZMat,-ViewPort.Dz);
    Mat_Multiply4x4(XMat,YMat);
    Mat_Multiply4x4(XMat,ZMat);
    For A:=0 to 2 do
      With Box[A] do
        Begin
          For B:=0 to TotalVertex do
            With VertexBank[B] do
              Begin
                CMat[0]:=XR;
                CMat[1]:=YR;
                CMat[2]:=ZR;
                CMat[3]:=1;
                Mat_Multiply4x1(XMat,CMat);
                X:=CMat[0];
                Y:=CMat[1];
                Z:=CMat[2];
              End;
          CMat[0]:=XR-ViewPort.X;
          CMat[1]:=YR-ViewPort.Y;
          CMat[2]:=ZR-ViewPort.Z;
          CMat[3]:=1;
          Mat_Multiply4x1(XMat,CMat);
          X:=CMat[0];
          Y:=CMat[1];
          Z:=CMat[2];
        End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
```

```
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].XR;
            CMat[1]:=VertexBank[d].YR;
            CMat[2]:=VertexBank[d].ZR;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].XR:=CMat[0];
            VertexBank[d].YR:=CMat[1];
            VertexBank[d].ZR:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    For A:=0 to 2 do
      Transform(A,XMat);
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    For A:=0 to 2 do
      Transform(A,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    For A:=0 to 2 do
      Transform(A,ZMat);
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 2 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 2 do
      Begin
        Box[A].XR:=Box[A].XR+Tx;
        Box[A].YR:=Box[A].YR+Ty;
        Box[A].ZR:=Box[A].ZR+Tz;
```

```
      End;
  End;

Procedure Painter_Sort;
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
    For A:=0 to ListCtr-2 do
     If PolygonList[A].active then
      For B:=A+1 to ListCtr-1 do
        If PolygonList[B].active then
          If PolygonList[A].ZMin<PolygonList[B].ZMin then
            Begin
              Bf:=PolygonList[A];
              PolygonList[A]:=PolygonList[B];
              PolygonList[B]:=Bf;
            End;
  End;

Procedure Draw_Map;
Var X,Y:Longint;
    A:Integer;
  Begin
    Rectangle(Buff^,10,10,110,100,15);
    Block(Buff^,11,11,109,99,0);
    Line(Buff^,60,50,60,60,14);
    Line(Buff^,55,55,65,55,14);
    X:=Round(ViewPort.X/20)+60;
    Y:=-Round(ViewPort.Z/20)+55;
    Rectangle(Screen^,X-2,Y-2,X+2,Y+2,30);
    Line(Screen^,X,Y,X+Round(Sinus(-ViewPort.Dy)*10),;
        Y-Round(Cosinus(ViewPort.Dy)*10),20);
    For A:=0 to 2 do
      Begin
        PutPixel(Buff^,Round(Box[A].XR/20)+60,;
         -Round(Box[A].ZR/20)+55,15);
      End;
    Rectangle(Buff^,10,105,110,195,15);
    Block(Buff^,11,106,109,194,0);
    Line(Buff^,60,145,60,155,14);
    Line(Buff^,55,150,65,150,14);
    X:=Round(ViewPort.X/20)+60;
    Y:=-Round(ViewPort.Y/20)+150;
    Rectangle(Screen^,X-2,Y-2,X+2,Y+2,30);
    Line(Screen^,X,Y,X+Round(Sinus(ViewPort.Dz)*10),;
        Y-Round(Cosinus(ViewPort.Dz)*10),20);
    For A:=0 to 2 do
      Begin
        PutPixel(Buff^,Round(Box[A].XR/20)+60,;
         -Round(Box[A].YR/20)+150,15);
      End;
  End;

Procedure Main;
Var Ch:Char;
  Begin
    Ch:='s';
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
```

```
        Remove_PolyList;
        ViewPort_Transform;
        Project_Object;
        View_Volume_Filter;
        BackFace_Filter;
        Painter_Sort;
        Display_PList_Shading;
        Draw_Map;
        Line(Buff^,160,0,160,199,15);
        Line(Buff^,0,100,319,100,15);
        MoveTo(Buff^,Screen^);
        Case ch of
          'q':X_Rotation(2);
          'a':X_Rotation(-2);
          'w':Y_Rotation(2);
          's':Y_Rotation(-2);
          'e':Z_Rotation(2);
          'd':Z_Rotation(-2);
          #80:ViewPort.Dx:=ViewPort.Dx+1;
          #72:ViewPort.Dx:=ViewPort.Dx-1;
          #77:ViewPort.Dy:=ViewPort.Dy+1;
          #75:ViewPort.Dy:=ViewPort.Dy-1;
          'y':ViewPort.Dz:=ViewPort.Dz+1;
          'h':ViewPort.Dz:=ViewPort.Dz-1;
          '4':ViewPort.X:=ViewPort.X-4;
          '6':ViewPort.X:=ViewPort.X+4;
          '8':ViewPort.Y:=ViewPort.Y+4;
          '2':ViewPort.Y:=ViewPort.Y-4;
          '-':ViewPort.Z:=ViewPort.Z+4;
          '+':ViewPort.Z:=ViewPort.Z-4;
        End;
      Until Keypressed;
      Ch:=Readkey;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

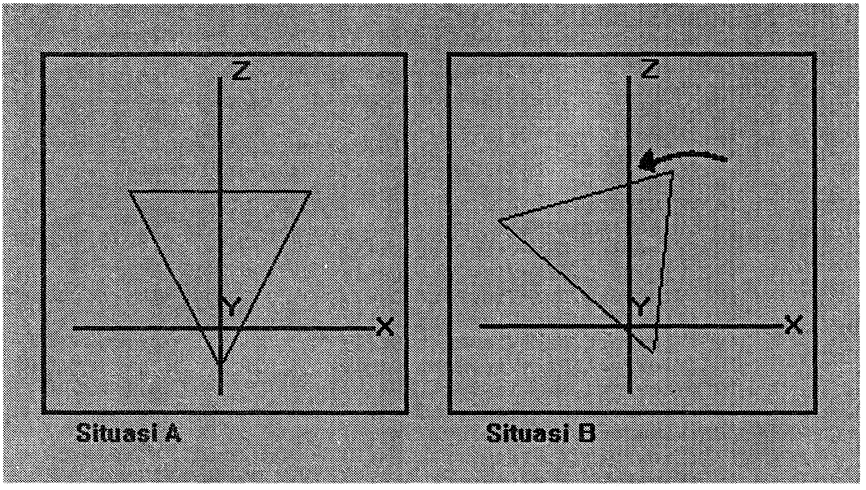
Gunakan tombol kursor untuk melakukan rotasi sumbu X dan Y dari viewport (pelihat) dan gunakan tombol y dan h untuk rotasi sumbu Z. Rotasi sumbu X ini dapat divisualisasikan sebagai pandangan orang yang sedang menengadah atau menunduk (Yaw). Sedangkan rotasi sumbu Y sebagai pandangan orang yang sedang menoleh ke kiri atau kanan (Pitch). Dan rotasi sumbu Z sebagai pandangan orang yang sedang memiringkan kepala (Roll).

Untuk menggeser posisi dari pelihat (viewport), coba gunakan tombol 2,8,6,4,- dan +.

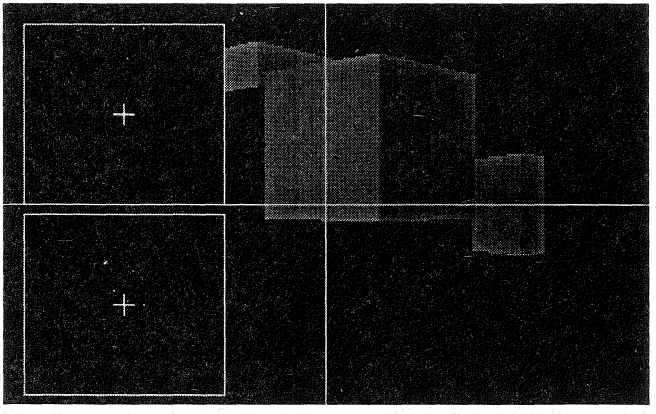
Pada sisi kiri layar terdapat dua buah jendela. Jendela pertama ialah peta

lokasi dari dunia tiga dimensi dilihat dari atas. Sedangkan pada jendela kedua ialah peta lokasi dari dunia tiga dimensi dilihat dari depan.

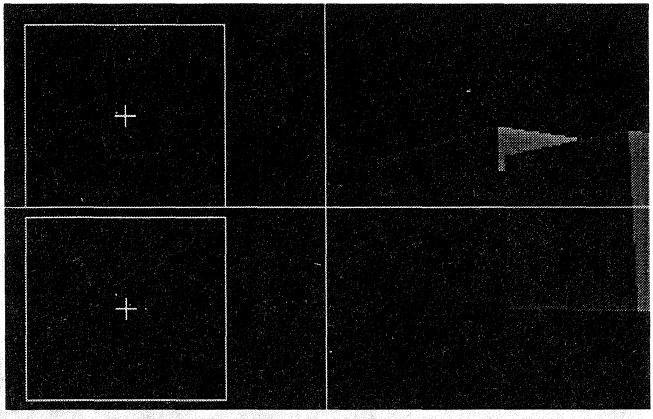
Tanda + pada masing-masing peta ialah koordinat (0,0,0) dari dunia tiga dimensi kita.



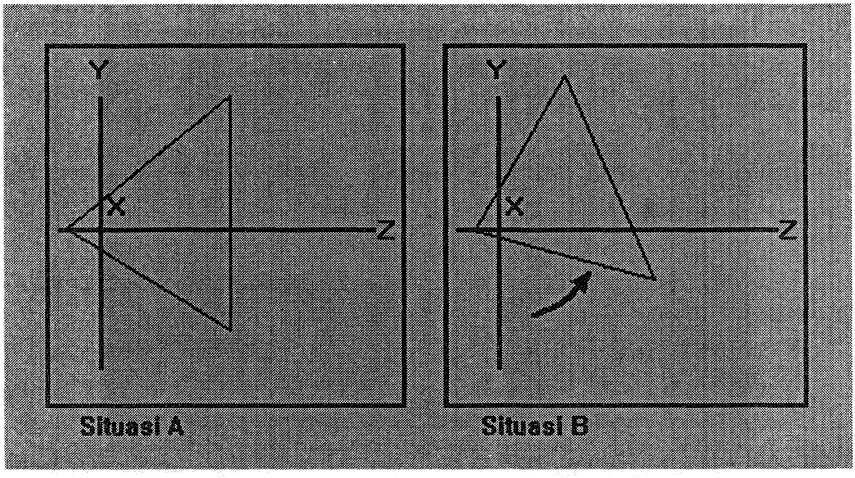
Perhatikan, pada saat dilakukan rotasi sumbu Y pada pelihat, ruang proyeksi ikut berputar. Sehingga objek-objek akan terlihat berputar ke kanan (arah kebalikannya).



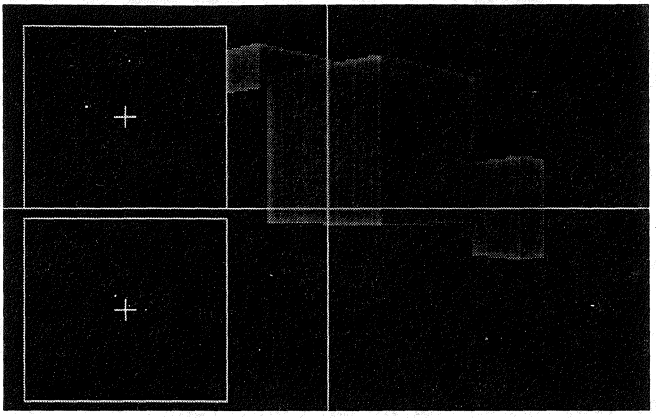
Pada saat ini viewport masih belum berputar.



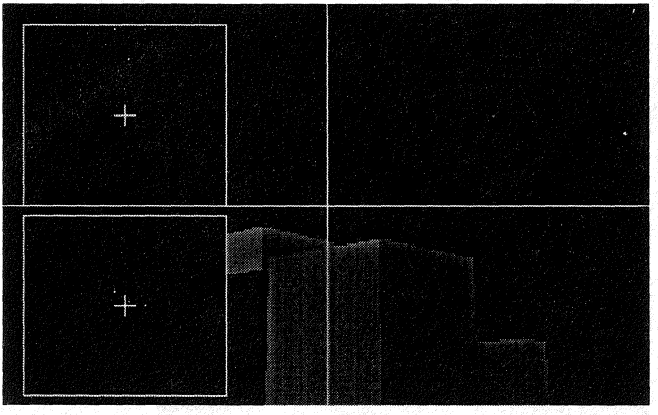
Pada saat viewport berputar ke kiri (rotasi terhadap sumbu Y), benda seakan-akan bergerak ke kanan.



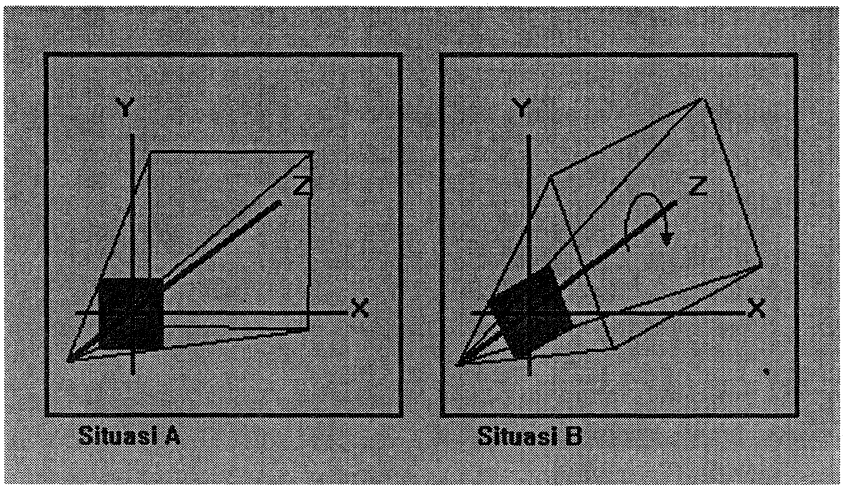
Demikian pula pada saat dilakukannya rotasi sumbu X. Ruang proyeksi berputar sehingga objek terlihat berputar ke bawah.



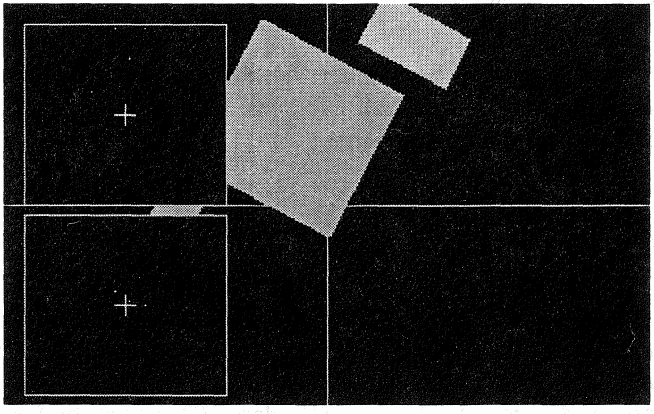
Viewport belum berputar.



Pada saat viewport berputar ke atas (rotasi terhadap sumbu X), benda terlihat berputar ke bawah.



Pada rotasi sumbu Z, objek-objek hasil proyeksi akan berputar pada sumbu Z berlawanan arah putaran pelihat.



Pada saat viewport berputar terhadap sumbu Z, benda terlihat berputar berlawanan arah putaran viewport.

PENJELASAN LISTING PROGRAM

Modifikasi yang pertama kita lakukan ialah pada bagian data structure dari vertex bank.

Di sini ditambahkan koordinat tiga dimensi XR, YR dan ZR yang akan berguna untuk menampung koordinat lokal tiga dimensi yang sesungguhnya dari sebuah titik. Sedangkan koordinat X, Y dan Z yang juga berada pada record ini berguna untuk menampung koordinat tiga dimensi semu hasil inverse transformasi dari titik.

```
Vertex_Rec = Record
       XR, YR, ZR : Real;
       X, Y, Z    : Real;
       Xp,Yp      : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..20] of Vertex_Rec;
       Polygon    : Array[0..20] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
```

X,Y dan Z berguna untuk menyimpan koordinat semu dari objek setelah dilakukan proses inverse transformasi.

```
       X,Y,Z      : Real;
```

Pada object data structure, ditambahkan XR, YR dan ZR yang berguna untuk menyimpan koordinat global dari objek.

```
       XR,YR,ZR   : Real;
     End;
```

Tambahan data structure baru ialah ViewPort_Rec yang berguna untuk menampung informasi dari pelihat. Informasi ini ialah koordinat global dari pelihat dan sudut pandang dari pelihat.

```
ViewPort_Rec = Record
       X, Y, Z  : Real;
       Dx,Dy,Dz : Real;
     End;
```

Adapun beberapa proses harus dimodifikasi. Yang pertama ialah pada saat kita merancang objek. Karena X, Y dan Z sekarang digunakan untuk menampung koordinat semu dari objek, maka koordinat lokal ini harus dimasukkan pada koordinat XR, YR dan ZR yang berfungsi untuk menampung koordinat lokal dari objek.

```
Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No].VertexBank[0].XR:=-lx;Box[No].VertexBank[0].YR:=ly;
        Box[No].VertexBank[0].ZR:=-lz;
    Box[No].VertexBank[1].XR:=lx;Box[No].VertexBank[1].YR:=ly;
        Box[No].VertexBank[1].ZR:=-lz;
    Box[No].VertexBank[2].XR:=lx;Box[No].VertexBank[2].YR:=-ly;
        Box[No].VertexBank[2].ZR:=-lz;
    Box[No].VertexBank[3].XR:=-lx;Box[No].VertexBank[3].YR:=-ly;
        Box[No].VertexBank[3].ZR:=-lz;
    Box[No].VertexBank[4].XR:=-lx;Box[No].VertexBank[4].YR:=ly;
        Box[No].VertexBank[4].ZR:=lz;
    Box[No].VertexBank[5].XR:=lx;Box[No].VertexBank[5].YR:=ly;
        Box[No].VertexBank[5].ZR:=lz;
```

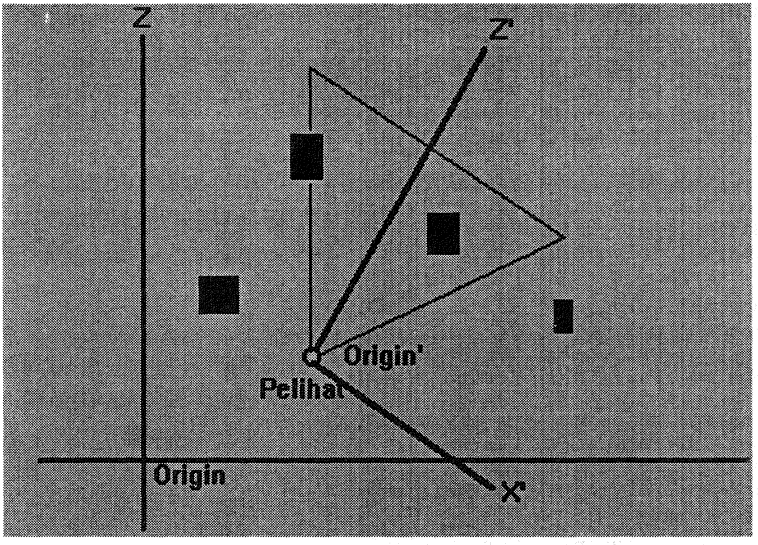
```
Box[No].VertexBank[6].XR:=lx;Box[No].VertexBank[6].YR:=-ly;
    Box[No].VertexBank[6].ZR:=lz;
Box[No].VertexBank[7].XR:=-lx;Box[No].VertexBank[7].YR:=-ly;
    Box[No].VertexBank[7].ZR:=lz;
Box[No].Polygon[0].TotalVertex:=4;
Box[No].Polygon[1].TotalVertex:=4;
Box[No].Polygon[2].TotalVertex:=4;
Box[No].Polygon[3].TotalVertex:=4;
Box[No].Polygon[4].TotalVertex:=4;
Box[No].Polygon[5].TotalVertex:=4;

Box[No].Polygon[0].Vertex[0]:=0;
Box[No].Polygon[0].Vertex[1]:=3;
Box[No].Polygon[0].Vertex[2]:=2;
Box[No].Polygon[0].Vertex[3]:=1;

Box[No].Polygon[1].Vertex[0]:=0;
Box[No].Polygon[1].Vertex[1]:=1;
Box[No].Polygon[1].Vertex[2]:=5;
Box[No].Polygon[1].Vertex[3]:=4;

Box[No].Polygon[2].Vertex[0]:=4;
Box[No].Polygon[2].Vertex[1]:=5;
Box[No].Polygon[2].Vertex[2]:=6;
Box[No].Polygon[2].Vertex[3]:=7;

Box[No].Polygon[3].Vertex[0]:=7;
Box[No].Polygon[3].Vertex[1]:=6;
Box[No].Polygon[3].Vertex[2]:=2;
Box[No].Polygon[3].Vertex[3]:=3;

Box[No].Polygon[4].Vertex[0]:=4;
Box[No].Polygon[4].Vertex[1]:=7;
Box[No].Polygon[4].Vertex[2]:=3;
Box[No].Polygon[4].Vertex[3]:=0;

Box[No].Polygon[5].Vertex[0]:=5;
Box[No].Polygon[5].Vertex[1]:=1;
Box[No].Polygon[5].Vertex[2]:=2;
Box[No].Polygon[5].Vertex[3]:=6;

Box[No].Polygon[0].Color:=17;
Box[No].Polygon[1].Color:=19;
Box[No].Polygon[2].Color:=21;
Box[No].Polygon[3].Color:=23;
Box[No].Polygon[4].Color:=25;
Box[No].Polygon[5].Color:=27;

Box[No].TotalPoly:=6;
Box[No].TotalVertex:=8;
```

Demikian pula dengan koordinat global object harus diletakkan pada XR, YR dan ZR.

```
    Box[No].XR:=X;Box[No].YR:=Y;Box[No].ZR:=Z;
End;
```

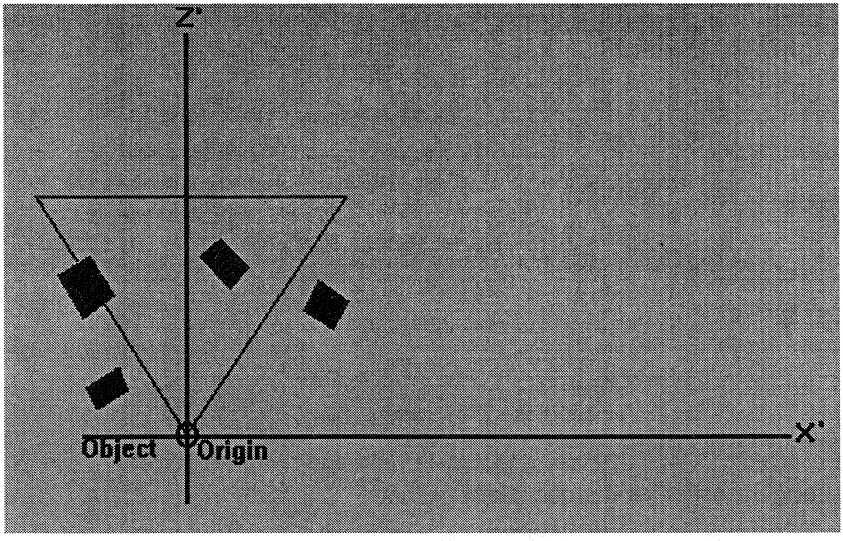
Nah procedure inilah yang memegang peranan terpenting dalam melakukan

inverse transform. Procedure ini dipanggil pertama kali sebelum dilakukan filter-filter. Kenapa? Seperti yang telah kita ketahui, salah satu filter ialah Volume Filter. Di sini dilakukan pengecekan apakah sebuah poligon di dalam view volume si pelihat atau tidak. Karena sekarang si pelihat mampu bergerak dan mengubah sudut pandangnya, maka ruang pandang/view volumenya juga ikut bergerak. Maka belum tentu sebuah objek yang setelah difilter dinyatakan di dalam ruang penglihatan, setelah pelihat bergerak objek tersebut masih berada di ruang penglihatan. Untuk itu filter harus dilakukan setelah dilakukan proses inverse transformasi.



Perhatikan diagram di atas. Di sini si pelihat telah bergeser dari origin dan telah berputar pada sumbu Y searah jarum jam. Dalam hal ini kita dapat membayangkan bahwa garis koordinat dari si pelihat ikut bergeser dan berotasi. Z' ialah koordinat frame Z yang baru dari pelihat. Sedangkan X' ialah koordinat frame X yang baru dari pelihat.

Untuk melakukan proyeksi, kita harus memproyeksikan objek terhadap koordinat frame dari pelihat, bukannya koordinat dunia. Dalam hal ini ialah X', Y' dan Z'.



Karena untuk memproyeksikan sebuah objek pada koordinat yang miring sangatlah sulit, maka dilakukan cara lain yaitu menggeser si pelihat ke titik origin dan merotasi sudut pandangnya sehingga seperti koordinat dunia. Dalam hal ini, penggeseran dan perotasian dilakukan pula terhadap seluruh objek dalam dunia tiga dimensi. Dan hasil inverse transformasi dari objek-objek tersebut disimpan di dalam koordinat semunya. Sehingga diperoleh situasi seperti gambar di atas yang merupakan hasil inverse transformasi dari diagram sebelumnya.

Setelah situasi ini diperoleh, langkah selanjutnya tinggal melakukan proyeksi seperti biasa.

Sederhana kan?

```
Procedure ViewPort_Transform;
Var A,B:Integer;
    CMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
```

Seperti yang telah kita pelajari. Untuk melakukan inverse transform, lakukan transformasi dengan kebalikan dari matrix transformasi.

```
    Mat_XRotate(XMat,-ViewPort.Dx);
    Mat_YRotate(YMat,-ViewPort.Dy);
    Mat_ZRotate(ZMat,-ViewPort.Dz);
    Mat_Multiply4x4(XMat,YMat);
    Mat_Multiply4x4(XMat,ZMat);
```

Lakukan scanning terhadap seluruh titik dalam seluruh objek.

```
    For A:=0 to 2 do
      With Box[A] do
        Begin
          For B:=0 to TotalVertex do
            With VertexBank[B] do
              Begin
                CMat[0]:=XR;
                CMat[1]:=YR;
                CMat[2]:=ZR;
                CMat[3]:=1;
                Mat_Multiply4x1(XMat,CMat);
                X:=CMat[0];
                Y:=CMat[1];
                Z:=CMat[2];
              End;
```

Kurangi lokasi objek dengan lokasi pelihat.

```
          CMat[0]:=XR-ViewPort.X;
          CMat[1]:=YR-ViewPort.Y;
          CMat[2]:=ZR-ViewPort.Z;
          CMat[3]:=1;
```

Lakukan inverse transformasi dari proses rotasi.

```
          Mat_Multiply4x1(XMat,CMat);
```

Masukkan koordinat semu hasil transformasi ke dalam X, Y dan Z.

Untuk selanjutnya koordinat semu inilah yang akan kita filter dan kita proyeksikan pada layar monitor.

```
          X:=CMat[0];
          Y:=CMat[1];
          Z:=CMat[2];
        End;
  End;
```

Fungsi di bawah ini berguna untuk melakukan lokal transformasi terhadap objek (memutar sebuah objek). Nah sekarang timbul pertanyaan. Mana yang harus ditransformasi. Koordinat semu atau koordinat asli. Benar!!! Koordinat asli, karena situasi yang terjadi sebenarnya ialah kita hendak memutar sebuah objek. Sehingga diperlukan proses terhadap koordinat aslinya.

```
Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
```

Lakukan scanning terhadap seluruh titik pada vertex bank.

```
For d:=0 to TotalVertex-1 do
  Begin
    CMat[0]:=VertexBank[d].XR;
    CMat[1]:=VertexBank[d].YR;
    CMat[2]:=VertexBank[d].ZR;
    CMat[3]:=1;
```

Kalikan koordinat asli dari objek tersebut dengan matrix transformasi A.

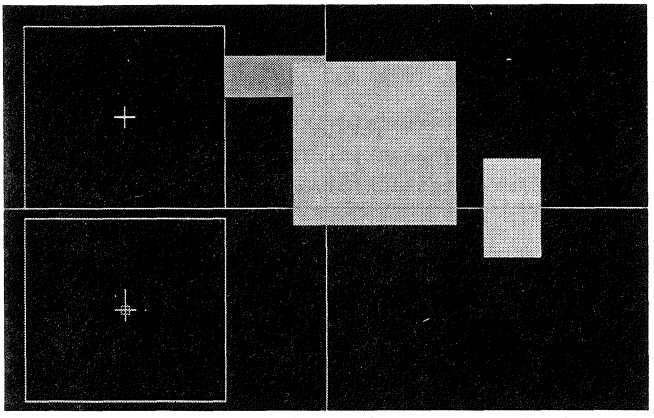
```
    Mat_Multiply4x1(A,CMat);
```

Update koordinat asli dari objek.

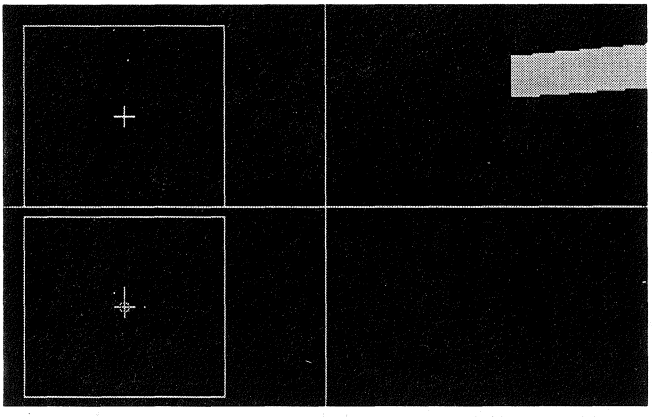
```
    VertexBank[d].XR:=CMat[0];
    VertexBank[d].YR:=CMat[1];
    VertexBank[d].ZR:=CMat[2];
  End;
End;
End;
```

8.2 Coordinate Frame

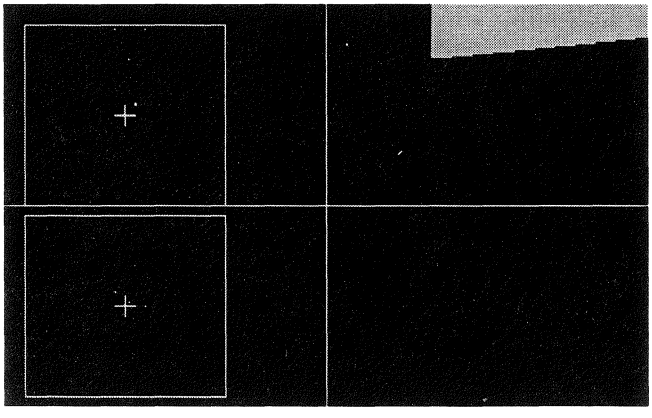
Jika Anda telah mencoba program B3VIEW.PAS kita di atas secara detil, akan terasa bahwa sangatlah sulit menggerakkan viewport (pelihat). Setelah menjalankan program untuk pertama kalinya, gerakkan pelihat ke depan dengan menekan tombol --- yang berarti menambah koordinat Z. Di sini terlihat viewport bergerak ke arah depan. Sekarang lakukan rotasi terhadap sumbu Y sekitar 90 derajat ke kiri dengan menekan tombol kursor kiri. Kemudian tekan lagi tombol --- untuk menambah koordinat Z. Lho!!! Kok viewport malahan bergeser ke kanan, bukannya bergeser ke depan seperti biasanya??? Apa yang terjadi di sini? Apa yang salah?



Viewport belum berputar.

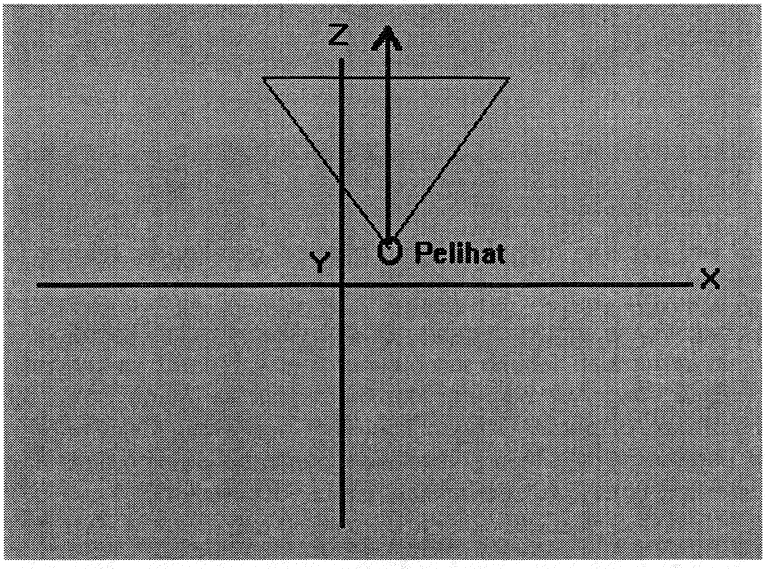


Viewport berputar ke arah kiri sehinga seakan-akan benda berputar ke kanan.

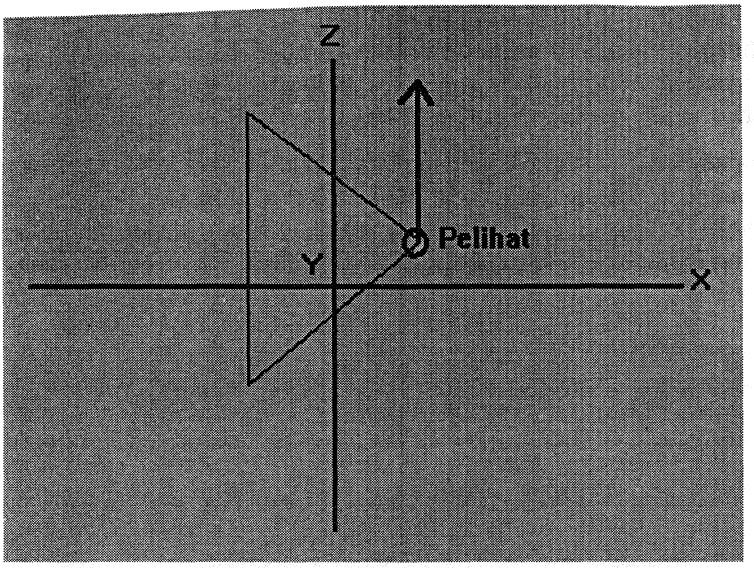


Pada saat tombol ditekan, kita tidak bergerak maju, malahan sepertinya kita bergerak ke arah kanan. Bandingkan posisi objek pada kanan atas layar pada gambar ini dan gambar sebelumnya. Terlihat objek bergerak ke kiri.

Nggak ada yang salah. Hal ini terjadi karena pada saat kita menekan tombol -, koordinat Z viewport bertambah. Dan pada saat kita telah berotasi 90 derajat ke kiri, sumbu koordinat Z itu tidak lagi di depan kita melainkan pada samping kanan kita. Sehingga gerakan kita bukannya ke depan, malahan ke kanan.

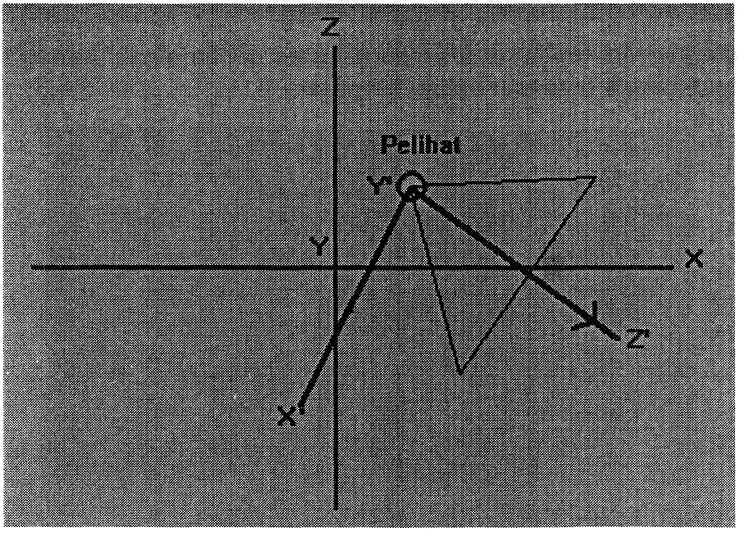


Pada diagram ini terlihat bahwa gerakan ke depan masih benar, karena sudut pandang kita masih belum berputar.



Nah, di sini gerakan maju ke depan menjadi gerakan ke kanan, karena sudut pandang kita telah menghadap ke kiri.

Lantas bagaimana pemecahannya. Di sini dikenal istilah koordinat frame. Koordinat frame ialah sebuah sumbu koordinat X, Y dan Z yang dimiliki oleh setiap objek yang ikut berputar dan berpindah ke mana objek tersebut berpindah dan berputar. Dengan koordinat frame ini, maka untuk melakukan pergeseran ke depan, dilakukan penambahan koordinat frame Z bukannya koordinat Z asli. Perhatikan gambar di bawah ini.



Nah, dengan cara ini maka objek terlihat bergerak maju. Sekarang yang menjadi masalah ialah, bagaimana mencari koordinat frame dari sebuah objek? Untuk mencarinya hampir tidak mungkin. Karena jika kita hanya mengalikan sudut X, Y dan Z dari objek dalam sebuah matrix akan terjadi kesalahan. Hal ini disebabkan oleh transformasi yang harus dilakukan secara berurutan. Sedangkan kita tidak memiliki informasi tentang urutan transformasi yang dilakukan pada sebuah objek. Kita hanya mengetahui hasil akhir DX, DY dan DZ. Transformasi Rx(10) >> Ry(20) >> Rx (10) >> Rz (10) >> Rz (30) tidak sama dengan transformasi Rx(20) >> Ry(20) >> Rz(40) yang didapat hanya dengan menjumlahkan total rotasi sumbu X, Y dan Z dan melakukannya sekaligus.

Bisa sih kalau kita menyimpan urutan rotasi tersebut dalam sebuah array. Dan setiap kali kita mau mencari koordinat frame, kita harus mengalikan seluruh transformasi tersebut secara berurutan. Cara ini benar-benar cara yang gawat!!! Selain memakan memori sangat besar karena setiap objek harus memiliki array transformasi, juga kecepatan animasi akan menjadi lambat. (Bahkan mungkin tidak akan bergerak kalau transformasi yang pernah dilakukan cukup banyak.) Terus bagaimana?

Sekali lagi matematika menjadi juru selamat bagi para pemrogram game. Jawabannya ialah dengan menggunakan matrix concatenation.

Kita dapat menyimpan seluruh proses transformasi yang pernah dilakukan pada sebuah objek dalam sebuah matrix. Sebuah matrix!!! Benar-benar cuma perlu sebuah matrix. WOW!!! Kita cukup mengalikan seluruh proses transformasi tersebut dalam sebuah matrix. Sehingga setiap kali kita melakukan sebuah transformasi baru pada objek, kita tidak langsung melakukan transformasi pada koordinat lokalnya, melainkan pada sebuah matrix koordinat frame yang disediakan pada setiap objek. Baru pada saat objek tersebut hendak diproyeksikan, dikalikanlah koordinat lokal objek tersebut dengan matrix koordinat frame tadi. Sehingga dalam hal ini, koordinat lokal dalam vertex bank tidak pernah berubah.

Dalam prakteknya dikenal dua buah sistem perkalian matrix yaitu pre-order dan post-order multiplication. Pre-order multiplication ialah mengalikan sebuah matrix baru pada matrix yang telah ada lewat sebelah kirinya. Hasilnya seperti contoh berikut.

$$A \times B = C$$

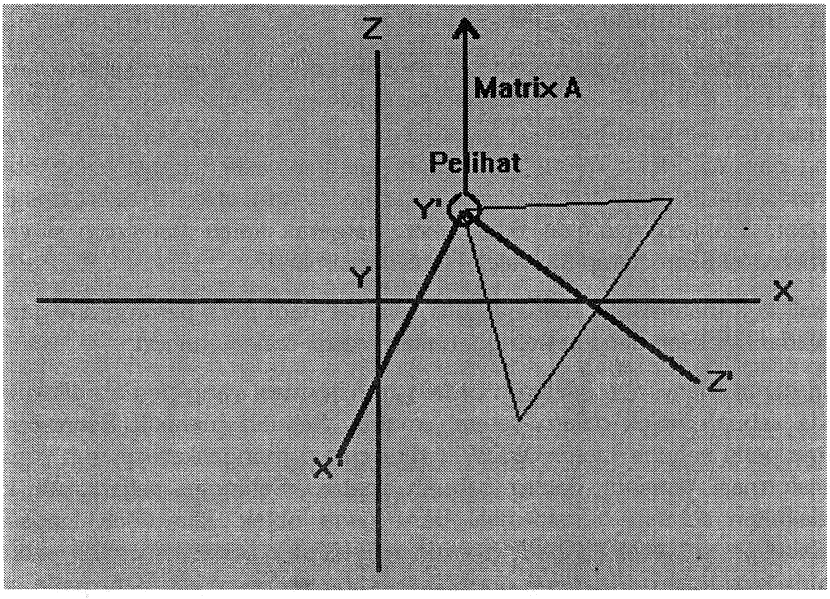
Sedangkan teknik kedua ialah sistem perkalian matrix post-order yang mengalikan sebuah matrix baru pada matrix yang telah ada lewat sebelah kanannya.

$$B \times A = D$$

Ingat A x B dengan B x A pada matrix akan menghasilkan jawaban yang lain.

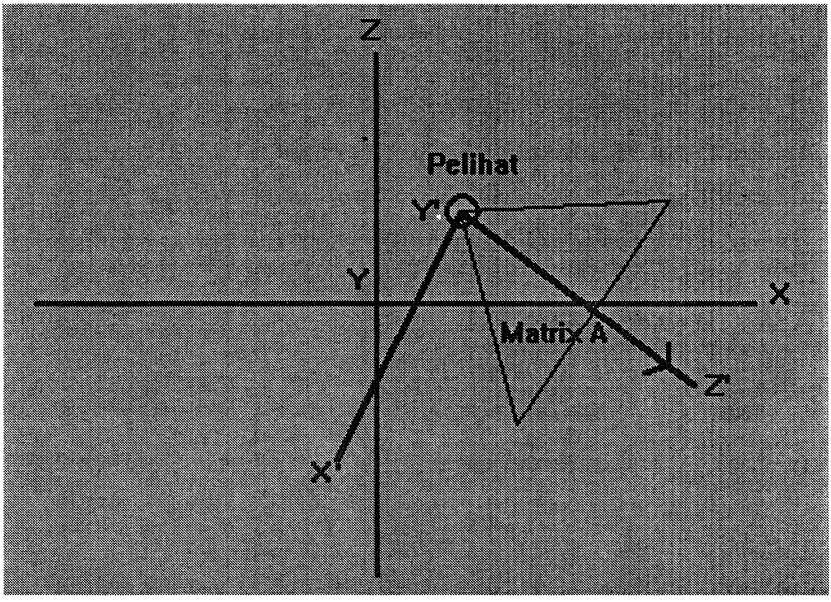
Apa kegunaan dua teknik ini?

Secara geometris, Pre-order multiplication antara matrix A (yang merupakan sebuah matrix transformasi baru yang hendak dilakukan pada sebuah objek) dengan matrix B (yang merupakan matrix koordinat frame yang berisi seluruh informasi transformasi yang pernah dilakukan terhadap objek tersebut) akan menghasilkan rotasi pada objek terhadap koordinat tiga dimensi asli (global koordinat) bukannya koordinat framenya.



Perhatikan. Seandainya matrix A yang hendak dikalikan ialah matrix dilatasi terhadap sumbu Z sejauh 10 maka dengan pre-order multiplication, objek akan berpindah sejauh 10 satuan ke arah sumbu Z dari dunia tiga dimensi asli.

Sedangkan dengan post-order multiplication, transformasi yang dilakukan relative terhadap koordinat frame yang terakhir dibentuk di dalam matrix koordinat frame.



Perhatikan gambar di atas. Matrix A yang sama dikalikan dengan post-order. Dihasilkan dilatasi ke arah sumbu Z' dari koordinat frame pada matrix B sejauh 10 satuan.

Hal ini tidak hanya berlaku untuk dilatasi saja lho, tetapi berlaku untuk seluruh transformasi matrix seperti scalling dan rotasi.

Masalah terpecahkan!!! Sekarang pelajari program di bawah ini.

```
{-----------------------------------------------                }
{ Program     : B3FLGHT.PAS                                    }
{ Coder       : Agustinus Nalwan                               }
{ Description: Program Flight Simulator dengan                 }
{              enam kebebasan gerak.                           }
{-----------------------------------------------                }

Uses Math,Crt,Dos,VgaCad;

Const View_Distance = 400;

Type
     Poly_Rec = Record
       Vertex      : Array[0..4] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       XR, YR, ZR : Real;
       X, Y, Z    : Real;
       Xp,Yp      : Longint;
     End;
```

```
    ObjRect = Record
      Color      : Byte;
      VertexBank : Array[0..80] of Vertex_Rec;
      Polygon    : Array[0..50] of Poly_Rec;
      TotalVertex: Integer;
      TotalPoly  : Integer;
      X,Y,Z      : Real;
      XR,YR,ZR   : Real;
      OTMat      : Mat4x4;
      OIMat      : Mat4x4;
    End;

    PolyList_Rec = Record
      ObjNo,PolyNo : Integer;
      Active       : Boolean;
      ZMin         : Real;
      ZMax         : Real;
      XMin         : Real;
      XMax         : Real;
      YMin         : Real;
      YMax         : Real;
      Color        : Integer;
    End;

    ViewPort_Rec = Record
      X, Y, Z   : Real;
      Dx,Dy,Dz : Real;
    End;

Var A,B:Integer;
    Box:array[0..10] of ObjRect;
    PolygonList:array[0..1000] of ^PolyList_Rec;
    ViewPort:ViewPort_Rec;
    ListCtr:Integer;
    buff:Pointer;
    VMat:Mat4x4;
    IMat:Mat4x4;
    ViewMode:Integer;
    CtrView,FixViewDistance:Longint;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
 Begin
    Box[No].VertexBank[0].XR:=-lx;Box[No].VertexBank[0].YR:=ly;
       Box[No].VertexBank[0].ZR:=-lz;
    Box[No].VertexBank[1].XR:=lx;Box[No].VertexBank[1].YR:=ly;
       Box[No].VertexBank[1].ZR:=-lz;
```

```
Box[No].VertexBank[2].XR:=lx;Box[No].VertexBank[2].YR:=-ly;
    Box[No].VertexBank[2].ZR:=-lz;
Box[No].VertexBank[3].XR:=-lx;Box[No].VertexBank[3].YR:=-ly;
    Box[No].VertexBank[3].ZR:=-lz;
Box[No].VertexBank[4].XR:=-lx;Box[No].VertexBank[4].YR:=ly;
    Box[No].VertexBank[4].ZR:=lz;
Box[No].VertexBank[5].XR:=lx;Box[No].VertexBank[5].YR:=ly;
    Box[No].VertexBank[5].ZR:=lz;
Box[No].VertexBank[6].XR:=lx;Box[No].VertexBank[6].YR:=-ly;
    Box[No].VertexBank[6].ZR:=lz;
Box[No].VertexBank[7].XR:=-lx;Box[No].VertexBank[7].YR:=-ly;
    Box[No].VertexBank[7].ZR:=lz;
Box[No].Polygon[0].TotalVertex:=4;
Box[No].Polygon[1].TotalVertex:=4;
Box[No].Polygon[2].TotalVertex:=4;
Box[No].Polygon[3].TotalVertex:=4;
Box[No].Polygon[4].TotalVertex:=4;
Box[No].Polygon[5].TotalVertex:=4;

Box[No].Polygon[0].Vertex[0]:=0;
Box[No].Polygon[0].Vertex[1]:=3;
Box[No].Polygon[0].Vertex[2]:=2;
Box[No].Polygon[0].Vertex[3]:=1;

Box[No].Polygon[1].Vertex[0]:=0;
Box[No].Polygon[1].Vertex[1]:=1;
Box[No].Polygon[1].Vertex[2]:=5;
Box[No].Polygon[1].Vertex[3]:=4;

Box[No].Polygon[2].Vertex[0]:=4;
Box[No].Polygon[2].Vertex[1]:=5;
Box[No].Polygon[2].Vertex[2]:=6;
Box[No].Polygon[2].Vertex[3]:=7;

Box[No].Polygon[3].Vertex[0]:=7;
Box[No].Polygon[3].Vertex[1]:=6;
Box[No].Polygon[3].Vertex[2]:=2;
Box[No].Polygon[3].Vertex[3]:=3;

Box[No].Polygon[4].Vertex[0]:=4;
Box[No].Polygon[4].Vertex[1]:=7;
Box[No].Polygon[4].Vertex[2]:=3;
Box[No].Polygon[4].Vertex[3]:=0;

Box[No].Polygon[5].Vertex[0]:=5;
Box[No].Polygon[5].Vertex[1]:=1;
Box[No].Polygon[5].Vertex[2]:=2;
Box[No].Polygon[5].Vertex[3]:=6;

Box[No].Polygon[0].Color:=Col;
Box[No].Polygon[1].Color:=Col;
Box[No].Polygon[2].Color:=Col;
Box[No].Polygon[3].Color:=Col;
Box[No].Polygon[4].Color:=Col;
Box[No].Polygon[5].Color:=Col;

Box[No].TotalPoly:=6;
Box[No].TotalVertex:=8;
```

```
    Box[No].XR:=X;Box[No].YR:=Y;Box[No].ZR:=Z;
  End;

Procedure SVB(ObjNo,VxNo:Integer;X,Y,Z:Real);
  Begin
    Box[ObjNo].VertexBank[VxNo].XR:=X;
    Box[ObjNo].VertexBank[VxNo].YR:=Y;
    Box[ObjNo].VertexBank[VxNo].ZR:=Z;
  End;

Procedure Create_Ship(No:Integer;X,Y,Z:Real);
Var Bf:Integer;
  Begin
    SVB(No,0,-100,3,0);
    SVB(No,1,-100,3,-10);
    SVB(No,2,-100,2,-10);
    SVB(No,3,-100,2,0);

    SVB(No,4,-60,2,5);
    SVB(No,5,-60,2,-10);
    SVB(No,6,-60,-2,-10);
    SVB(No,7,-60,-2,5);

    SVB(No,8,-40,5,13);
    SVB(No,9,-40,5,-10);
    SVB(No,10,-40,-2,-10);
    SVB(No,11,-40,-2,13);

    SVB(No,12,-40,9,55);
    SVB(No,13,-15,9,55);
    SVB(No,14,-15,-7,45);
    SVB(No,15,-40,-7,45);

    SVB(No,16,-40,9,-10);
    SVB(No,17,-20,9,-10);
    SVB(No,18,-20,-7,-15);
    SVB(No,19,-40,-7,-15);

    SVB(No,20,-30,25,-8);
    SVB(No,21,-30,25,-10);
    SVB(No,22,-31,9,-10);
    SVB(No,23,-31,9,0);
    SVB(No,24,-29,9,0);
    SVB(No,25,-29,9,-10);

    SVB(No,52,-18,0,35);
    SVB(No,53,18,0,35);
    SVB(No,54,-10,15,10);
    SVB(No,55,10,15,10);

    SVB(No,56,-10,5,0);
    SVB(No,57,10,5,0);
    SVB(No,58,-18,0,0);
    SVB(No,59,18,0,0);

    For A:=0 to 25 do
      Begin
```

```
    Box[No].VertexBank[26+A]:=Box[No].VertexBank[A];
    Box[No].VertexBank[26+A].XR:=-Box[No].VertexBank[26+A].XR;
  End;

For A:=0 to 60 do
  Begin
    Box[No].Polygon[A].TotalVertex:=4;
  End;

Box[No].Polygon[18].TotalVertex:=3;
Box[No].Polygon[19].TotalVertex:=3;
Box[No].Polygon[38].TotalVertex:=3;
Box[No].Polygon[39].TotalVertex:=3;

Box[No].Polygon[0].Vertex[0]:=7;
Box[No].Polygon[0].Vertex[1]:=11;
Box[No].Polygon[0].Vertex[2]:=10;
Box[No].Polygon[0].Vertex[3]:=6;

Box[No].Polygon[1].Vertex[0]:=0;
Box[No].Polygon[1].Vertex[1]:=3;
Box[No].Polygon[1].Vertex[2]:=2;
Box[No].Polygon[1].Vertex[3]:=1;

Box[No].Polygon[2].Vertex[0]:=0;
Box[No].Polygon[2].Vertex[1]:=1;
Box[No].Polygon[2].Vertex[2]:=5;
Box[No].Polygon[2].Vertex[3]:=4;

Box[No].Polygon[3].Vertex[0]:=4;
Box[No].Polygon[3].Vertex[1]:=5;
Box[No].Polygon[3].Vertex[2]:=9;
Box[No].Polygon[3].Vertex[3]:=8;

Box[No].Polygon[4].Vertex[0]:=1;
Box[No].Polygon[4].Vertex[1]:=2;
Box[No].Polygon[4].Vertex[2]:=6;
Box[No].Polygon[4].Vertex[3]:=5;

Box[No].Polygon[5].Vertex[0]:=5;
Box[No].Polygon[5].Vertex[1]:=6;
Box[No].Polygon[5].Vertex[2]:=10;
Box[No].Polygon[5].Vertex[3]:=9;

Box[No].Polygon[6].Vertex[0]:=8;
Box[No].Polygon[6].Vertex[1]:=9;
Box[No].Polygon[6].Vertex[2]:=10;
Box[No].Polygon[6].Vertex[3]:=11;

Box[No].Polygon[7].Vertex[0]:=0;
Box[No].Polygon[7].Vertex[1]:=4;
Box[No].Polygon[7].Vertex[2]:=7;
Box[No].Polygon[7].Vertex[3]:=3;

Box[No].Polygon[8].Vertex[0]:=4;
Box[No].Polygon[8].Vertex[1]:=8;
Box[No].Polygon[8].Vertex[2]:=11;
```

```
Box[No].Polygon[8].Vertex[3]:=7;

Box[No].Polygon[9].Vertex[0]:=3;
Box[No].Polygon[9].Vertex[1]:=7;
Box[No].Polygon[9].Vertex[2]:=6;
Box[No].Polygon[9].Vertex[3]:=2;

Box[No].Polygon[10].Vertex[0]:=12;
Box[No].Polygon[10].Vertex[1]:=15;
Box[No].Polygon[10].Vertex[2]:=19;
Box[No].Polygon[10].Vertex[3]:=16;

Box[No].Polygon[11].Vertex[0]:=12;
Box[No].Polygon[11].Vertex[1]:=16;
Box[No].Polygon[11].Vertex[2]:=17;
Box[No].Polygon[11].Vertex[3]:=13;

Box[No].Polygon[12].Vertex[0]:=12;
Box[No].Polygon[12].Vertex[1]:=13;
Box[No].Polygon[12].Vertex[2]:=14;
Box[No].Polygon[12].Vertex[3]:=15;

Box[No].Polygon[13].Vertex[0]:=13;
Box[No].Polygon[13].Vertex[1]:=17;
Box[No].Polygon[13].Vertex[2]:=18;
Box[No].Polygon[13].Vertex[3]:=14;

Box[No].Polygon[14].Vertex[0]:=15;
Box[No].Polygon[14].Vertex[1]:=14;
Box[No].Polygon[14].Vertex[2]:=18;
Box[No].Polygon[14].Vertex[3]:=19;

Box[No].Polygon[15].Vertex[0]:=16;
Box[No].Polygon[15].Vertex[1]:=19;
Box[No].Polygon[15].Vertex[2]:=18;
Box[No].Polygon[15].Vertex[3]:=17;

Box[No].Polygon[16].Vertex[0]:=20;
Box[No].Polygon[16].Vertex[1]:=23;
Box[No].Polygon[16].Vertex[2]:=22;
Box[No].Polygon[16].Vertex[3]:=21;

Box[No].Polygon[17].Vertex[0]:=20;
Box[No].Polygon[17].Vertex[1]:=21;
Box[No].Polygon[17].Vertex[2]:=25;
Box[No].Polygon[17].Vertex[3]:=24;

Box[No].Polygon[18].Vertex[0]:=20;
Box[No].Polygon[18].Vertex[1]:=24;
Box[No].Polygon[18].Vertex[2]:=23;

Box[No].Polygon[19].Vertex[0]:=21;
Box[No].Polygon[19].Vertex[1]:=22;
Box[No].Polygon[19].Vertex[2]:=25;
```

```
Box[No].Polygon[20].Vertex[0]:=7;
Box[No].Polygon[20].Vertex[1]:=11;
Box[No].Polygon[20].Vertex[2]:=10;
Box[No].Polygon[20].Vertex[3]:=6;

Box[No].Polygon[21].Vertex[0]:=0;
Box[No].Polygon[21].Vertex[1]:=3;
Box[No].Polygon[21].Vertex[2]:=2;
Box[No].Polygon[21].Vertex[3]:=1;

Box[No].Polygon[22].Vertex[0]:=0;
Box[No].Polygon[22].Vertex[1]:=1;
Box[No].Polygon[22].Vertex[2]:=5;
Box[No].Polygon[22].Vertex[3]:=4;

Box[No].Polygon[23].Vertex[0]:=4;
Box[No].Polygon[23].Vertex[1]:=5;
Box[No].Polygon[23].Vertex[2]:=9;
Box[No].Polygon[23].Vertex[3]:=8;

Box[No].Polygon[24].Vertex[0]:=1;
Box[No].Polygon[24].Vertex[1]:=2;
Box[No].Polygon[24].Vertex[2]:=6;
Box[No].Polygon[24].Vertex[3]:=5;

Box[No].Polygon[25].Vertex[0]:=5;
Box[No].Polygon[25].Vertex[1]:=6;
Box[No].Polygon[25].Vertex[2]:=10;
Box[No].Polygon[25].Vertex[3]:=9;

Box[No].Polygon[26].Vertex[0]:=8;
Box[No].Polygon[26].Vertex[1]:=9;
Box[No].Polygon[26].Vertex[2]:=10;
Box[No].Polygon[26].Vertex[3]:=11;

Box[No].Polygon[27].Vertex[0]:=0;
Box[No].Polygon[27].Vertex[1]:=4;
Box[No].Polygon[27].Vertex[2]:=7;
Box[No].Polygon[27].Vertex[3]:=3;

Box[No].Polygon[28].Vertex[0]:=4;
Box[No].Polygon[28].Vertex[1]:=8;
Box[No].Polygon[28].Vertex[2]:=11;
Box[No].Polygon[28].Vertex[3]:=7;

Box[No].Polygon[29].Vertex[0]:=3;
Box[No].Polygon[29].Vertex[1]:=7;
Box[No].Polygon[29].Vertex[2]:=6;
Box[No].Polygon[29].Vertex[3]:=2;

Box[No].Polygon[30].Vertex[0]:=12;
Box[No].Polygon[30].Vertex[1]:=15;
Box[No].Polygon[30].Vertex[2]:=19;
Box[No].Polygon[30].Vertex[3]:=16;

Box[No].Polygon[31].Vertex[0]:=12;
Box[No].Polygon[31].Vertex[1]:=16;
```

```
Box[No].Polygon[31].Vertex[2]:=17;
Box[No].Polygon[31].Vertex[3]:=13;

Box[No].Polygon[32].Vertex[0]:=12;
Box[No].Polygon[32].Vertex[1]:=13;
Box[No].Polygon[32].Vertex[2]:=14;
Box[No].Polygon[32].Vertex[3]:=15;

Box[No].Polygon[33].Vertex[0]:=13;
Box[No].Polygon[33].Vertex[1]:=17;
Box[No].Polygon[33].Vertex[2]:=18;
Box[No].Polygon[33].Vertex[3]:=14;

Box[No].Polygon[34].Vertex[0]:=15;
Box[No].Polygon[34].Vertex[1]:=14;
Box[No].Polygon[34].Vertex[2]:=18;
Box[No].Polygon[34].Vertex[3]:=19;

Box[No].Polygon[35].Vertex[0]:=16;
Box[No].Polygon[35].Vertex[1]:=19;
Box[No].Polygon[35].Vertex[2]:=18;
Box[No].Polygon[35].Vertex[3]:=17;

Box[No].Polygon[36].Vertex[0]:=20;
Box[No].Polygon[36].Vertex[1]:=23;
Box[No].Polygon[36].Vertex[2]:=22;
Box[No].Polygon[36].Vertex[3]:=21;

Box[No].Polygon[37].Vertex[0]:=20;
Box[No].Polygon[37].Vertex[1]:=21;
Box[No].Polygon[37].Vertex[2]:=25;
Box[No].Polygon[37].Vertex[3]:=24;

Box[No].Polygon[38].Vertex[0]:=20;
Box[No].Polygon[38].Vertex[1]:=24;
Box[No].Polygon[38].Vertex[2]:=23;

Box[No].Polygon[39].Vertex[0]:=21;
Box[No].Polygon[39].Vertex[1]:=22;
Box[No].Polygon[39].Vertex[2]:=25;

Box[No].Polygon[40].Vertex[0]:=52;
Box[No].Polygon[40].Vertex[1]:=54;
Box[No].Polygon[40].Vertex[2]:=55;
Box[No].Polygon[40].Vertex[3]:=53;

Box[No].Polygon[41].Vertex[0]:=54;
Box[No].Polygon[41].Vertex[1]:=56;
Box[No].Polygon[41].Vertex[2]:=57;
Box[No].Polygon[41].Vertex[3]:=55;

Box[No].Polygon[42].Vertex[0]:=56;
Box[No].Polygon[42].Vertex[1]:=58;
Box[No].Polygon[42].Vertex[2]:=59;
Box[No].Polygon[42].Vertex[3]:=57;
```

```
    Box[No].Polygon[43].Vertex[0]:=52;
    Box[No].Polygon[43].Vertex[1]:=58;
    Box[No].Polygon[43].Vertex[2]:=56;
    Box[No].Polygon[43].Vertex[3]:=54;

    Box[No].Polygon[44].Vertex[0]:=53;
    Box[No].Polygon[44].Vertex[1]:=55;
    Box[No].Polygon[44].Vertex[2]:=57;
    Box[No].Polygon[44].Vertex[3]:=59;

    Box[No].Polygon[45].Vertex[0]:=52;
    Box[No].Polygon[45].Vertex[1]:=53;
    Box[No].Polygon[45].Vertex[2]:=59;
    Box[No].Polygon[45].Vertex[3]:=58;

    For A:=20 to 39 do
      For B:=0 to Box[No].Polygon[A].TotalVertex-1 do
        Begin
          Inc(Box[No].Polygon[A].Vertex[B],26);
        End;

    For A:=20 to 39 do
      For B:=0 to (Box[No].Polygon[A].TotalVertex-1) div 2 do
        Begin
          Bf:=Box[No].Polygon[A].Vertex[B];
          Box[No].Polygon[A].Vertex[B]:=Box[No].Polygon[A].Vertex[
            Box[No].Polygon[A].TotalVertex-1-B];
          Box[No].Polygon[A].Vertex[Box[No].Polygon[A].;
              TotalVertex-1-B]:=Bf;
        End;

    For A:=0 to 45 do Box[No].Polygon[A].Color:=120;
    For A:=10 to 15 do Box[No].Polygon[A].Color:=40;
    For A:=16 to 19 do Box[No].Polygon[A].Color:=0;
    For A:=36 to 39 do Box[No].Polygon[A].Color:=0;
    For A:=30 to 35 do Box[No].Polygon[A].Color:=40;
    For A:=41 to 45 do Box[No].Polygon[A].Color:=0;
    Box[No].Polygon[41].Color:=40;
    Box[No].TotalPoly:=46;
    Box[No].TotalVertex:=60;
    Box[No].XR:=X;Box[No].YR:=Y;Box[No].ZR:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 100 do
      PolygonList[A]^.Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Randomize;
    For A:=1 to 10 do
```

```
      Begin
        Mat_Identity(Box[A].OTMat);
        Mat_Identity(Box[A].OIMat);
       Create_Box(A,Round(Random(600))-300,;
        Round(Random(600))-300,;
           Round(Random(1000))+500,;
                Round(Random(40))+30,Round(Random(40))+50,
          Round(Random(50))+50,((A mod 5)-1)*40);
      End;
    Mat_Identity(Box[0].OTMat);
    Mat_Identity(Box[0].OIMat);
    Create_Ship(0,0,0,300);
    Remove_PolyList;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A,63-Round(A*1.5),63-Round(A*1.5),;
            63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+40,63-Round(A*1.5),0,0);
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+80,0,63-Round(A*1.5),0);
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+120,0,0,63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+160,0,63-Round(A*1.5),63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+200,63-Round(A*1.5),63-Round(A*1.5),0);
      End;
    For A:=0 to 1000 do GetMem(PolygonList[A],SizeOf(PolyList_Rec));

    With ViewPort do
      Begin
        X:=0;Y:=0;Z:=0;Dx:=0;Dy:=0;Dz:=0;
      End;
    ViewMode:=0;
    Mat_Identity(VMat);
    Mat_Identity(IMat);
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalVertex-1 do
```

```
        Begin
          Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
            VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
        End;
    End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No] do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr]^.Color:=Polygon[A].Color;
            PolygonList[ListCtr]^.ObjNo:=No;
            PolygonList[ListCtr]^.PolyNo:=A;
            PolygonList[ListCtr]^.Active:=True;
            PolygonList[ListCtr]^.ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr]^.ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr]^.XMax:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr]^.XMin:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr]^.YMax:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            PolygonList[ListCtr]^.YMin:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr]^.ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                   PolygonList[ListCtr]^.ZMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr]^.ZMin>VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                   PolygonList[ListCtr]^.ZMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;

                If PolygonList[ListCtr]^.YMax<VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                   PolygonList[ListCtr]^.YMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr]^.YMin>VertexBank;
                  [Polygon[A].Vertex[B]].Yp then
                   PolygonList[ListCtr]^.YMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Yp;
                If PolygonList[ListCtr]^.XMax<VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                   PolygonList[ListCtr]^.XMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Xp;
                If PolygonList[ListCtr]^.XMin>VertexBank;
                  [Polygon[A].Vertex[B]].Xp then
                   PolygonList[ListCtr]^.XMin:=VertexBank;
                       [Polygon[A].Vertex[B]].Xp;
              End;
            Inc(ListCtr);
          End;
```

```
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A]^.Active then
      Begin
        ObjNo :=PolygonList[A]^.ObjNo;
        PolyNo:=PolygonList[A]^.PolyNo;
        With Box[ObjNo] do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
                X1:=X2;          Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo].Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo,Color:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
    YCtr        : Longint;
    YMin,YMax   : Longint;
    YMinIndex   : Integer;
    YMaxIndex   : Integer;
    RightCtr    : Integer;
    LeftCtr     : Integer;
    XL,XR,Bf    : Longint;
    Dx,Dy       : Real;
    Finish      : Boolean;
    Ty          : Longint;
  Begin
    YMin:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMinIndex:=0;
    YMax:=Box[ObjNo].VertexBank[Poly.Vertex[0]].Yp;
    YMaxIndex:=0;

    For A:=1 to Poly.TotalVertex-1 do
      Begin
        Ty:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        If YMin>Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
          Begin
            YMin:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
```

```
        YMinIndex:=A;
      End;
    If YMax<Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp then
      Begin
        YMax:=Box[ObjNo].VertexBank[Poly.Vertex[A]].Yp;
        YMaxIndex:=A;
      End;
  End;

LeftStart:=YMinIndex; RightStart:=YMinIndex;
YCtr :=YMin;
LeftEnd:=LeftStart+1;
If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
RightEnd:=RightStart-1;
If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
   Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
   Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
Finish:=False;
While (Dy=0) and (Not Finish) do
  Begin
    LeftStart:=LeftStart+1;
    If LeftStart>=Poly.TotalVertex then LeftStart:=0;
    LeftEnd:=LeftStart+1;
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
      Xp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[LeftEnd]].;
      Yp-Box[ObjNo].VertexBank[Poly.Vertex[LeftStart]].Yp+1;
    If LeftStart=YMaxIndex then Finish:=True;
  End;
If Not Finish then
  Begin
    LeftSlope:=(Dx)/(Dy);
    Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
      Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
    Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
      Yp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Yp+1;
    While (Dy=0) and (Not Finish) do
      Begin
        RightStart:=RightStart-1;
        If RightStart<0 then RightStart:=Poly.TotalVertex-1;
        RightEnd:=RightStart-1;
        Dx:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
           Xp-Box[ObjNo].VertexBank[Poly.Vertex[RightStart]].Xp;
        Dy:=Box[ObjNo].VertexBank[Poly.Vertex[RightEnd]].;
           Yp-Box[ObjNo].VertexBank;
            [Poly.Vertex[RightStart]].Yp+1;
        If RightStart=YMaxIndex then Finish:=True;
      End;
    RightSlope:=(Dx)/(Dy);
    XLeft :=Box[ObjNo].VertexBank;
      [Poly.Vertex[LeftStart]].Xp+LeftSlope;
    XRight:=Box[ObjNo].VertexBank;
      [Poly.Vertex[RightStart]].Xp+RightSlope;
    While (Not Finish) do
      Begin
        XL:=Round(XLeft);
        XR:=Round(XRight);
        If XL>XR then Finish:=True;
        If XL<0 then XL:=0;
```

```
If XR>319 then XR:=319;
If (YCtr>=0) and (Not Finish) then
 If XL=XR then else
  If XL<XR then
    HLines(Buff^,XL,XR,YCtr,Color);
XLeft:=XLeft+LeftSlope;
XRight:=XRight+RightSlope;
Inc(YCtr);
If YCtr>199 then Finish:=True;
If Not Finish then
 If (YCtr>Box[ObjNo].VertexBank;
   [Poly.Vertex[LeftEnd]].Yp) then
  Begin
    Repeat
      LeftStart:=LeftStart+1;
      If LeftStart>=Poly.TotalVertex then LeftStart:=0;
      LeftEnd:=LeftStart+1;
      If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
      Dx:=Box[ObjNo].VertexBank;
        [Poly.Vertex[LeftEnd]].Xp-Box[ObjNo].VertexBank;
          [Poly.Vertex[LeftStart]].Xp;
      Dy:=Box[ObjNo].VertexBank;
        [Poly.Vertex[LeftEnd]].Yp-Box[ObjNo].VertexBank;
          [Poly.Vertex[LeftStart]].Yp+1;
      If LeftStart=YMaxIndex then Finish:=True;
    Until (Dy<>0) or Finish;
    If Not Finish then
      Begin
        LeftSlope:=(Dx)/(Dy);
        XLeft:=Box[ObjNo].VertexBank;
          [Poly.Vertex[LeftStart]].Xp+LeftSlope;
      End;
  End;
If Not Finish then
  Begin
    If YCtr>Box[ObjNo].VertexBank;
      [Poly.Vertex[RightEnd]].Yp then
      Begin
        Repeat
          RightStart:=RightStart-1;
          If RightStart<0 then;
            RightStart:=Poly.TotalVertex-1;
          RightEnd:=RightStart-1;
          If RightEnd<0 then;
            RightEnd:=Poly.TotalVertex-1;
          Dx:=Box[ObjNo].VertexBank;
            [Poly.Vertex[RightEnd]].Xp-Box[ObjNo].;
              VertexBank[Poly.Vertex[RightStart]].Xp;
          Dy:=Box[ObjNo].VertexBank;
            [Poly.Vertex[RightEnd]].Yp-Box[ObjNo].;
              VertexBank[Poly.Vertex[RightStart]].Yp+1;
          If RightStart=YMaxIndex then Finish:=True;
        Until (Dy<>0) or Finish;
        If Not Finish then
          Begin
            RightSlope:=(Dx)/(Dy);
            XRight:=Box[ObjNo].VertexBank;
                [Poly.Vertex[RightStart]].Xp;
            XRight:=XRight+RightSlope;
          End;
      End;
```

```
            End;
          End;
      End;
  End;

Function Vertical_VV_Testing(YMin,YMax:Real) : Boolean;
  Begin
    Vertical_VV_Testing:=False;
    If (YMin>199) or (YMax<0) then Vertical_VV_Testing:=True;
  End;

Function Horisontal_VV_Testing(XMin,XMax:Real) : Boolean;
  Begin
    Horisontal_VV_Testing:=False;
    If (XMin>319) or (XMax<0) then Horisontal_VV_Testing:=True;
  End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    ColCtr:=17;
    For A:=0 to ListCtr-1 do
     If PolygonList[A]^.Active then
      Begin
        ObjNo :=PolygonList[A]^.ObjNo;
        PolyNo:=PolygonList[A]^.PolyNo;
        Poly_Shading(Box[ObjNo].Polygon;
          [PolyNo],ObjNo,PolygonList[A]^.Color);
      End;
  End;

Procedure View_Volume_Filter;
Var A,B:Integer;
  Begin
    For A:=0 to ListCtr-1 do
      If PolygonList[A]^.Active then
        Begin
          If PolygonList[A]^.ZMax<=100 then;
             PolygonList[A]^.Active:=False;
          If PolygonList[A]^.ZMin<=50  then;
             PolygonList[A]^.Active:=False;
          If Vertical_VV_Testing;
             (PolygonList[A]^.YMin,PolygonList[A]^.YMax)
            then PolygonList[A]^.Active:=False;
          If Horisontal_VV_Testing;
            (PolygonList[A]^.XMin,PolygonList[A]^.XMax)
            then PolygonList[A]^.Active:=False;
        End;
  End;

Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
    Deg             :Real;
    Intensity       :Integer;
  Begin
    View[0]:=0;View[1]:=0;View[2]:=10;
    For Ctr:=0 to ListCtr-1 do
      Begin
```

```
        ObjNo :=PolygonList[Ctr]^.ObjNo;
        PolyNo:=PolygonList[Ctr]^.PolyNo;
        A[0]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[0]].X-
               Box[ObjNo].VertexBank;
                 [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        A[1]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Y-
               Box[ObjNo].VertexBank;
                 [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        A[2]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[0]].Z-
               Box[ObjNo].VertexBank;
                 [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;

        View[0]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        View[1]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        View[2]:=-Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z-View_Distance;

        B[0]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[2]].X-
               Box[ObjNo].VertexBank;
                 [Box[ObjNo].Polygon[PolyNo].Vertex[1]].X;
        B[1]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Y-
               Box[ObjNo].VertexBank;
                 [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Y;
        B[2]:=Box[ObjNo].VertexBank;
         [Box[ObjNo].Polygon[PolyNo].Vertex[2]].Z-
               Box[ObjNo].VertexBank;
                 [Box[ObjNo].Polygon[PolyNo].Vertex[1]].Z;
        Vect_Cross(Normal,A,B);
        Deg:=Abs(Abs(Vect_Angle(Normal,View)));
        Intensity:=Round(Deg/2.5);
        PolygonList[Ctr]^.Color:=Box[ObjNo].Polygon;
         [PolygonList[Ctr]^.PolyNo].Color+Intensity+1;
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to 10 do
      Begin
        VertexBank_Projection(A);
        Assign_Polygon_List(A);
      End;
  End;

Procedure ViewPort_Transform;
Var A,B:Integer;
    CMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    For A:=0 to 10 do
      With Box[A] do
        Begin
```

```
          For B:=0 to TotalVertex do
            With VertexBank[B] do
              Begin
                CMat[0]:=XR;
                CMat[1]:=YR;
                CMat[2]:=ZR;
                CMat[3]:=1;
                Mat_Multiply4x1(OTMat,CMat);
                Mat_Multiply4x1(IMat,CMat);
                X:=CMat[0];
                Y:=CMat[1];
                Z:=CMat[2];
              End;
          CMat[0]:=XR-ViewPort.X;
          CMat[1]:=YR-ViewPort.Y;
          CMat[2]:=ZR-ViewPort.Z;
          CMat[3]:=1;
          Mat_Multiply4x1(IMat,CMat);
          X:=CMat[0];
          Y:=CMat[1];
          Z:=CMat[2];
        End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No] do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].XR;
            CMat[1]:=VertexBank[d].YR;
            CMat[2]:=VertexBank[d].ZR;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].XR:=CMat[0];
            VertexBank[d].YR:=CMat[1];
            VertexBank[d].ZR:=CMat[2];
          End;
      End;
  End;

Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
    Mat_Multiply4x4(Box[0].OTMat,XMat);
    Mat_XRotate(XMat,-Deg);
    Mat_Multiply4x4(XMat,Box[0].OIMat);
    Box[0].OIMat:=XMat;
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
```

```
    Mat_Multiply4x4(Box[0].OTMat,YMat);
    Mat_YRotate(YMat,-Deg);
    Mat_Multiply4x4(YMat,Box[0].OIMat);
    Box[0].OIMat:=YMat;
  End;

Procedure Y_Rotation2(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
    Mat_Multiply4x4(YMat,Box[0].OTMat);
    Box[0].OTMat:=YMat;
    Mat_YRotate(YMat,-Deg);
    Mat_Multiply4x4(Box[0].OIMat,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
    Mat_Multiply4x4(Box[0].OTMat,ZMat);
    Mat_ZRotate(ZMat,-Deg);
    Mat_Multiply4x4(ZMat,Box[0].OIMat);
    Box[0].OIMat:=ZMat;
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to 10 do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
  Begin
    For A:=0 to 10 do
      Begin
        Box[A].XR:=Box[A].XR+Tx;
        Box[A].YR:=Box[A].YR+Ty;
        Box[A].ZR:=Box[A].ZR+Tz;
      End;
  End;

Procedure Painter_Sort;
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
    For A:=0 to ListCtr-2 do
     If PolygonList[A]^.active then
      For B:=A+1 to ListCtr-1 do
        If PolygonList[B]^.active then
          If PolygonList[A]^.ZMin<PolygonList[B]^.ZMin then
            Begin
              Bf:=PolygonList[A]^;
```

```
              PolygonList[A]^:=PolygonList[B]^;
              PolygonList[B]^:=Bf;
            End;
  End;

Procedure Draw_Map;
Var X,Y:Longint;
    A:Integer;
  Begin
    Rectangle(Buff^,10,10,110,100,15);
    Block(Buff^,11,11,109,99,0);
    Line(Buff^,60,50,60,60,14);
    Line(Buff^,55,55;65,55,14);
    X:=Round(ViewPort.X/20)+60;
    Y:=-Round(ViewPort.Z/20)+55;
    Rectangle(Screen^,X-2,Y-2,X+2,Y+2,30);
    For A:=0 to 10 do
      Begin
        PutPixel(Buff^,Round(Box[A].XR/20)+60,;
          -Round(Box[A].ZR/20)+55,15);
      End;
    Rectangle(Buff^,10,105,110,195,15);
    Block(Buff^,11,106,109,194,0);
    Line(Buff^,60,145,60,155,14);
    Line(Buff^,55,150,65,150,14);
    X:=Round(ViewPort.X/20)+60;
    Y:=-Round(ViewPort.Y/20)+150;
    Rectangle(Screen^,X-2,Y-2,X+2,Y+2,30);
    For A:=0 to 10 do
      Begin
        PutPixel(Buff^,Round(Box[A].XR/20)+60,;
          -Round(Box[A].YR/20)+150,15);
      End;
  End;

Procedure Move_Object(Dir:Integer);
Var BMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    BMat[0]:=0;
    BMat[1]:=0;
    BMat[2]:=4*Dir;
    BMat[3]:=1;
    Mat_Multiply4x1(Box[0].OTMat,BMat);
    Box[0].XR:=Box[0].XR+BMat[0];
    Box[0].YR:=Box[0].YR+BMat[1];
    Box[0].ZR:=Box[0].ZR+BMat[2];
  End;

Procedure Adjust_ViewPort(No:Integer);
Var BMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    If ViewMode<>0 then
      Begin
        If CtrView<FixViewDistance then
          Begin
            CtrView:=CtrView+20;
            If CtrView>FixViewdistance then;
               CtrView:=FixViewdistance;
```

```
        End;
      If CtrView>FixViewDistance then
        Begin
          CtrView:=CtrView-20;
          If CtrView<FixViewdistance then;
             CtrView:=FixViewdistance;
        End;
      BMat[0]:=0;
      BMat[1]:=20;
      BMat[2]:=-CtrView;
      BMat[3]:=1;
      Mat_Multiply4x1(Box[0].OTMat,BMat);
      ViewPort.X:=Box[0].XR+BMat[0];
      ViewPort.Y:=Box[0].YR+BMat[1];
      ViewPort.Z:=Box[0].ZR+BMat[2];
      VMat:=Box[0].OTMat;
      IMat:=Box[0].OIMat;
    End;
  End;

Procedure Main;
Var Ch:Char;
    Speed:Integer;
    DZ:Integer;
    DY:Integer;
    DX:Integer;
  Begin
    Ch:='s';
    Speed:=0;
    DZ:=0;
    DY:=0;
    DX:=0;
    CtrView:=50;
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        ViewPort_Transform;
        Project_Object;
        View_Volume_Filter;
        BackFace_Filter;
        Painter_Sort;
        Display_PList_Shading;
        MoveTo(Buff^,Screen^);
        Case ch of
          #80:Begin
                If DX<30 then
                  Begin
                    X_Rotation(4);
                    Inc(DX,4);
                  End;
              End;
          #72:Begin
                If DX>-30 then
                  Begin
                    X_Rotation(-4);
                    Dec(DX,4);
                  End;
              End;
          #75:Begin
                If DZ>-30 then
```

```
        Begin
          Z_Rotation(-4);
          Dec(DZ,4);
        End;
    End;
#77:Begin
      If DZ<30 then
        Begin
          Z_Rotation(4);
          Inc(DZ,4);
        End;
    End;
'r':X_Rotation(2);
'f':X_Rotation(-2);
'a':Speed:=2;
'z':Speed:=-2;
' ':Begin
      If DX<0 then
        Begin
          If DX>-4 then
            Begin
              X_Rotation(-DX);
              DX:=0;
            End Else
            Begin
              X_Rotation(4);
              Inc(DX,4);
            End;
        End;
      If DX>0 then
        Begin
          If DX<4 then
            Begin
              X_Rotation(-DX);
              DX:=0;
            End Else
            Begin
              X_Rotation(-4);
              Dec(DX,4);
            End;
        End;
      If DZ<0 then
        Begin
          If DZ>-3 then
            Begin
              Z_Rotation(-DZ);
              DZ:=0;
            End Else
            Begin
              Z_Rotation(3);
              Inc(DZ,3);
            End;
        End;
      If DZ>0 then
        Begin
          If DZ<3 then
            Begin
              Z_Rotation(-DZ);
              DZ:=0;
            End Else
            Begin
```

```
                    Z_Rotation(-3);
                    Dec(DZ,3);
                  End;
                End;
              End;
          End;
          If DZ<0 then
            Begin
              Y_Rotation2(2);
            End;
          If DZ>0 then
            Begin
              Y_Rotation2(-2);
            End;
          Move_Object(Speed);
          Adjust_ViewPort(ViewMode);
        Until Keypressed;
        Ch:=Readkey;
        if Ch=#13 then Inc(ViewMode);
        FixViewdistance:=ViewMode*100-50;
        If ViewMode>4 then ViewMode:=0;
      Until Ch=#27;
    End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

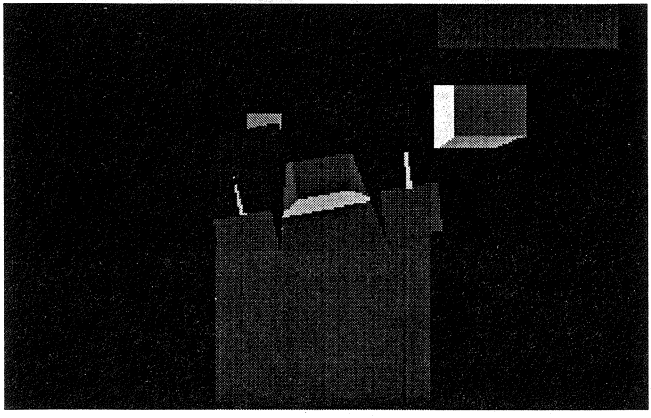
PENJELASAN PROGRAM

Coba jalankan program di atas. WOW!!! Flight Simulator. Gunakan tombol berikut untuk mengontrol pesawat.

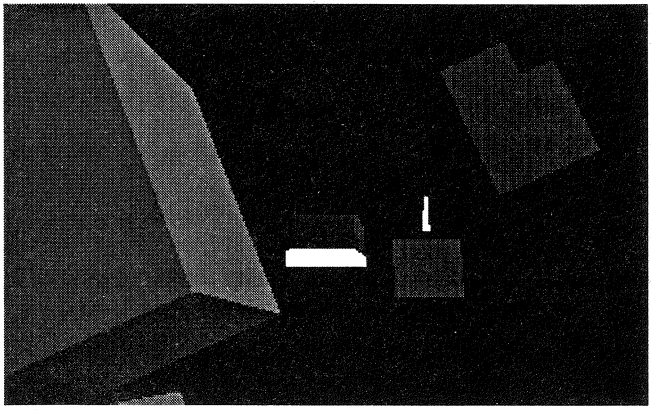
Kursor Kiri	Memiringkan sayap kiri pesawat sambil berputar ke kiri.
Kursor Kanan	Memiringkan sayap kanan pesawat sambil berputar ke kanan.
Kursor Atas	Menukik ke bawah.
Kursor Bawah	Menanjak ke atas.
Space Bar	Mengembalikan arah pesawat ke posisi normal.
Tombol A	Maju ke depan.
Tombol Z	Mundur ke belakang.
Enter	Mengganti sudut pandang. Ada 4 macam sudut pandang yang diberikan.

Cobalah menggerak-gerakkan pesawat di dunia tiga dimensi tersebut. Tekan Enter untuk mengganti sudut pandang. Sudut pandang pertama ialah kamera (viewport) dalam posisi diam. Tekan Enter sekali lagi untuk memasuki mode pandang kokpit (di dalam pesawat). Tekan Enter lagi untuk sudut pandang belakang pesawat. Tekan Enter beberapa kali lagi untuk menjauhkan kamera dari belakang pesawat. Dan terakhir Anda akan kembali

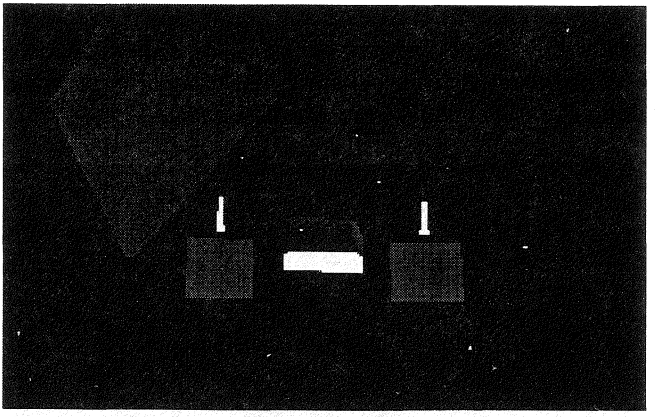
ke mode pertama. WOW!!! Seperti pada game DAYTONA nih!!! Sekarang rotasikan pesawat Anda ke segala penjuru, kemudian jalankan pesawat ke depan. WOW!!! Gerakkan maju sekarang telah benar-benar ke arah depan sudut pandang kita. Tidak lagi ke arah sumbu Z asli.



Camera berada pada sudut pandang diam.



Mode viewport dua sedang dipilih, pesawat sedang melintasi objek-objek.



Masih pada mode viewport 2 di mana kamera mengekor pada belakang pesawat.

PENJELASAN LISTING PROGRAM

Di sini dilakukan sedikit modifikasi pada program kita sebelum-sebelumnya. Seperti biasa, modifikasi pertama ialah pada data structure.

```
ObjRect = Record
        Color       : Byte;
        VertexBank  : Array[0..80] of Vertex_Rec;
        Polygon     : Array[0..50] of Poly_Rec;
        TotalVertex: Integer;
        TotalPoly   : Integer;
        X,Y,Z       : Real;
        XR,YR,ZR    : Real;
```

Seperti yang telah kita pelajari. Pada setiap objek ditambahkan sebuah matrix yaitu matrix koordinat frame dari objek tersebut.

Matrix ini akan menyimpan seluruh informasi transformasi secara berurutan yang dilakukan pada sebuah objek.

```
        OTMat       : Mat4x4;
```

Matrix kedua ialah matrix inverse dari transformasi. Matrix ini berisi inverse transform dari matrix di atas.

```
        OIMat       : Mat4x4;
      End;
```

Modifikasi berikutnya ialah pada variabel global.

```
Var A,B:Integer;
    Box:array[0..10] of ObjRect;
    PolygonList:array[0..1000] of ^PolyList_Rec;
```

```
    ViewPort:ViewPort_Rec;
    ListCtr:Integer;
    buff:Pointer;
```

Ditambahkan dua buah matrix yaitu VMat sebagai matrix koordinat frame dari camera (viewport).

Kita anggap camera adalah sebuah objek yang memiliki koordinat frame juga.

```
    VMat:Mat4x4;
```

Seperti biasa, untuk keperluan inverse transformasi dari viewport, digunakan matrix inverse.

```
    IMat:Mat4x4;
```

Jenis View yang digunakan (pandangan diam, pandangan belakang pesawat, pandangan kokpit dan sebagainya).

```
    ViewMode:Integer;
```

CtrView ialah jarak antara camera dengan titik tengah pesawat yang kita kendalikan.

FixViewDistance ialah jarak antara camera dengan titik tengah pesawat yang sedang dituju. Dengan ini dihasilkan efek camera yang bergerak perlahan menjauhi atau mendekati pesawat yang sedang digerakkan.

```
    CtrView,FixViewDistance:Longint;

Procedure SVB(ObjNo,VxNo:Integer;X,Y,Z:Real);
  Begin
    Box[ObjNo].VertexBank[VxNo].XR:=X;
    Box[ObjNo].VertexBank[VxNo].YR:=Y;
    Box[ObjNo].VertexBank[VxNo].ZR:=Z;
  End;
```

Procedure di bawah ini digunakan untuk merancang pesawat yang kita gerakkan. Procedure ini cukup panjang dan sebaiknya dalam pembuatan game-game tiga dimensi, dalam merancang objek dilakukan dengan editor. (Dalam hal ini Anda dapat menggunakan editor yang telah ada yaitu 3D Studio dengan mempelajari format file 3DS-nya untuk dibaca lewat program Anda, juga dapat dilakukan dengan menciptakan 3D EDITOR sendiri.) Perancangan objek lewat program sangatlah riskan dan memakan waktu lama. Di samping objek yang dihasilkan kurang bagus, karena kita tidak dapat membayangkan objek tersebut pada saat kita rancang.

Karena prosesnya yang sangat sederhana, yaitu seperti pada Create_Box maka procedure ini akan kita tinggalkan.

```
Procedure Create_Ship(No:Integer;X,Y,Z:Real);
Var Bf:Integer;
  Begin
    SVB(No,0,-100,3,0);
    SVB(No,1,-100,3,-10);
    SVB(No,2,-100,2,-10);
    SVB(No,3,-100,2,0);

    SVB(No,4,-60,2,5);
    SVB(No,5,-60,2,-10);
    SVB(No,6,-60,-2,-10);
    SVB(No,7,-60,-2,5);

    SVB(No,8,-40,5,13);
    SVB(No,9,-40,5,-10);
    SVB(No,10,-40,-2,-10);
    SVB(No,11,-40,-2,13);

    SVB(No,12,-40,9,55);
    SVB(No,13,-15,9,55);
    SVB(No,14,-15,-7,45);
    SVB(No,15,-40,-7,45);

    SVB(No,16,-40,9,-10);
    SVB(No,17,-20,9,-10);
    SVB(No,18,-20,-7,-15);
    SVB(No,19,-40,-7,-15);

    SVB(No,20,-30,25,-8);
    SVB(No,21,-30,25,-10);
    SVB(No,22,-31,9,-10);
    SVB(No,23,-31,9,0);
    SVB(No,24,-29,9,0);
    SVB(No,25,-29,9,-10);

    SVB(No,52,-18,0,35);
    SVB(No,53,18,0,35);
    SVB(No,54,-10,15,10);
    SVB(No,55,10,15,10);

    SVB(No,56,-10,5,0);
    SVB(No,57,10,5,0);
    SVB(No,58,-18,0,0);
    SVB(No,59,18,0,0);
.........................
```

Pembenahan berikutnya ialah pada procedure Initialize.

```
Procedure Initialize;
Var A:Integer;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Randomize;
```

Ciptakan 10 buah kotak pada dunia tiga dimensi.

```
    For A:=1 to 10 do
      Begin
```

Isi matrix koordinat fram dan inversenya dari setiap objek dengan matrix identitas, karena seluruh koordinat frame dari objek masih sama seperti koordinat asli.

```
        Mat_Identity(Box[A].OTMat);
        Mat_Identity(Box[A].OIMat);
        Create_Box(A,Round(Random(600))-300,Round(Random(600))-300,
          Round(Random(1000))+500,Round(Random(40))+30,;
             Round(Random(40))+50,
          Round(Random(50))+50,((A mod 5)-1)*40);
      End;
    Mat_Identity(Box[0].OTMat);
    Mat_Identity(Box[0].OIMat);
```

Panggil Create_Ship untuk menciptakan pesawat yang kita gerakkan.

```
    Create_Ship(0,0,0,300);
    Remove_PolyList;
```

Di sini dilakukan perhitungan palette. Palette nomor 1 hingga 40 diisi dengan warna mulai putih terang hingga hitam. Nomor-nomor palette berikutnya diisi dengan warna hijau, biru, kuning, merah dan lain-lain yang masing-masingnya memiliki 40 intensitas warna. Hal ini ditujukan agar kita dapat memakai beberapa warna selain putih untuk merancang objek. Dan lagi warna-warna tersebut memiliki intensitas untuk keperluan flat shadded light sourcing.

```
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A,63-Round(A*1.5),63-Round(A*1.5),;
            63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+40,63-Round(A*1.5),0,0);
      End;
    For A:=1 to 40 do
      Begin
```

```
      SetVisualPalette(A+80,0,63-Round(A*1.5),0);
    End;
  For A:=1 to 40 do
    Begin
      SetVisualPalette(A+120,0,0,63-Round(A*1.5));
    End;
  For A:=1 to 40 do
    Begin
      SetVisualPalette(A+160,0,63-Round(A*1.5),63-Round(A*1.5));
    End;
  For A:=1 to 40 do
    Begin
      SetVisualPalette(A+200,63-Round(A*1.5),63-Round(A*1.5),0);
    End;
  For A:=0 to 1000 do GetMem(PolygonList[A],SizeOf(PolyList_Rec));

  With ViewPort do
    Begin
      X:=0;Y:=0;Z:=0;Dx:=0;Dy:=0;Dz:=0;
    End;
```

Set View Mode ke mode 0 yaitu camera tidak bergerak.

```
  ViewMode:=0;
```

Deklarasikan matrix koordinat frame dari camera dan inversenya sebagai matrix identitas karena belum dilakukannya transformasi.

```
  Mat_Identity(VMat);
  Mat_Identity(IMat);
End;
```

Pada procedure ini dilakukan sedikit modifikasi.

```
Procedure ViewPort_Transform;
Var A,B:Integer;
  CMat:Mat1x4;
  XMat,YMat,ZMat:Mat4x4;
Begin
```

Lakukan scanning terhadap seluruh objek.

```
  For A:=0 to 10 do
    With Box[A] do
      Begin
```

Lakukan scanning terhadap seluruh titik di dalam vertex bank dari setiap objek.

```
        For B:=0 to TotalVertex do
          With VertexBank[B] do
```

```
Begin
  CMat[0]:=XR;
  CMat[1]:=YR;
  CMat[2]:=ZR;
  CMat[3]:=1;
```

Transformasikan koordinat titik dengan koordinat framenya yang disimpan pada matrix OTMat untuk mendapatkan koordinat titik yang sesungguhnya.

```
Mat_Multiply4x1(OTMat,CMat);
```

Transformasikan koordinat titik yang sesungguhnya dengan inverse transform dari camera untuk mentransformasikannya relative terhadap sudut pandang dan posisi camera.

Proses ini menggantikan proses perkalian antara matrix XMat, YMat dan ZMat yang masing-masing diisi dengan sudut -DX, -DY dan -DZ seperti yang biasa kita lakukan sebelumnya. Karena kamera merupakan sebuah objek, maka dia pun memiliki koordinat frame.

```
Mat_Multiply4x1(IMat,CMat);
```

Simpan koordinat hasil seluruh transformasi ke dalam koordinat semu.

```
  X:=CMat[0];
  Y:=CMat[1];
  Z:=CMat[2];
End;
```

Transformasikan pula koordinat global dari objek. Karena koordinat global dari objek selalu berada pada koordinat (0,0,0) relative terhadap lokal koordinat sistemnya, maka tidak perlu dilakukan transformasi terhadap koordinat framenya. Cukup dilakukan transformasi terhadap inverse transform dari camera.

```
      CMat[0]:=XR-ViewPort.X;
      CMat[1]:=YR-ViewPort.Y;
      CMat[2]:=ZR-ViewPort.Z;
      CMat[3]:=1;
      Mat_Multiply4x1(IMat,CMat);
      X:=CMat[0];
      Y:=CMat[1];
      Z:=CMat[2];
    End;
End;
```

Procedure di bawah ini berguna untuk melakukan rotasi terhadap sumbu X terhadap objek pertama yaitu pesawat yang kita gerakkan.

```
Procedure X_Rotation(Deg:Real);
Var XMat:Mat4x4;
```

```
    A:Integer;
  Begin
    Mat_XRotate(XMat,Deg);
```

Perhatikan. Dilakukan perkalian post-order terhadap matrix koordinat frame dari objek dengan matrix transformasi yang baru. Sehingga dalam hal ini rotasi dilakukan berdasarkan koordinat frame dari objek.

```
    Mat_Multiply4x4(Box[0].OTMat,XMat);
    Mat_XRotate(XMat,-Deg);
```

Sebagai lawannya. Dalam menghitung inverse. Kalikan matrix inverse dengan rotasi sumbu X dengan arah berlawanan dengan metode pre-order. Dengan cara ini inverse matrix juga dibentuk dari hasil perkalian beberapa matrix.

```
    Mat_Multiply4x4(XMat,Box[0].OIMat);
    Box[0].OIMat:=XMat;
  End;

Procedure Y_Rotation(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
```

Perhatikan. Dilakukan perkalian post-order terhadap matrix koordinat frame dari objek dengan matrix transformasi yang baru. Sehingga dalam hal ini rotasi dilakukan berdasarkan koordinat frame dari objek.

```
    Mat_Multiply4x4(Box[0].OTMat,YMat);
    Mat_YRotate(YMat,-Deg);
```

Sebagai lawannya. Dalam menghitung inverse. Kalikan matrix inverse dengan rotasi sumbu Y dengan arah berlawanan dengan metode pre-order. Dengan cara ini inverse matrix juga dibentuk dari hasil perkalian beberapa matrix.

```
    Mat_Multiply4x4(YMat,Box[0].OIMat);
    Box[0].OIMat:=YMat;
  End;

Procedure Y_Rotation2(Deg:Real);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Mat_YRotate(YMat,Deg);
```

Di sini dilakukan transformasi rotasi sumbu Y dengan metode pre-order. Tujuannya ialah melakukan rotasi sumbu Y pada objek terhadap koordinat

yang asli.

```
    Mat_Multiply4x4(YMat,Box[0].OTMat);
    Box[0].OTMat:=YMat;
    Mat_YRotate(YMat,-Deg);
```

Untuk inversenya, kita gunakan post-order dengan arah rotasi yang terbalik.

```
    Mat_Multiply4x4(Box[0].OIMat,YMat);
  End;

Procedure Z_Rotation(Deg:Real);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Mat_ZRotate(ZMat,Deg);
```

Perhatikan. Dilakukan perkalian post-order terhadap matrix koordinat frame dari objek dengan matrix transformasi yang baru. Sehingga dalam hal ini rotasi dilakukan berdasarkan koordinat frame dari objek.

```
    Mat_Multiply4x4(Box[0].OTMat,ZMat);
    Mat_ZRotate(ZMat,-Deg);
```

Sebagai lawannya. Dalam menghitung inverse. Kalikan matrix inverse dengan rotasi sumbu Z dengan arah berlawanan dengan metode pre-order. Dengan cara ini inverse matrix juga dibentuk dari hasil perkalian beberapa matrix.

```
    Mat_Multiply4x4(ZMat,Box[0].OIMat);
    Box[0].OIMat:=ZMat;
  End;
```

Procedure ini berguna untuk menggerakkan objek pertama yaitu pesawat kita ke arah depan atau ke arah belakang.

```
Procedure Move_Object(Dir:Integer);
Var BMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
```

Karena untuk menggerakkan objek ke depan atau ke belakang dilakukan dengan menambah atau mengurangi koordinat frame Z dari objek (bukan koordinat Z asli lho). Untuk itu kita perlu melakukan transformasi sebuah vektor yang merupakan vektor dilatasi ke arah sumbu Z asli. Vektor ini ditransformasikan dengan matrix koordinat frame dari objek sehingga vektor ini yang semula adalah dilatasi ke arah sumbu Z asli, sekarang menjadi vektor dilatasi ke arah sumbu Z koordinat frame dari objek. Untuk menentukan arah dilatasi maka pergeseran sumbu Z dikalikan dengan parameter DIR yang berisi -1 untuk pergeseran mundur dan berisi 1 untuk pergeseran maju.

```
    BMat[0]:=0;
    BMat[1]:=0;
    BMat[2]:=4*Dir;
    BMat[3]:=1;
```

Kalikan vektor BMat dengan matrix koordinat frame dari objek.

```
    Mat_Multiply4x1(Box[0].OTMat,BMat);
```

Hasilnya ialah sebuah vektor dilatasi yang relative terhadap sumbu koordinat frame Z dari objek. Sehingga, kita tinggal menambahkan masing-masing komponen vektor ini ke koordinat dari tiap objek yang bersangkutan.

```
    Box[0].XR:=Box[0].XR+BMat[0];
    Box[0].YR:=Box[0].YR+BMat[1];
    Box[0].ZR:=Box[0].ZR+BMat[2];
  End;
```

Procedure Adjust_ViewPort berfungsi sebagai pengatur mode viewport yang kita pilih. Juga melakukan perhitungan di mana kamera harus diletakkan dan ke mana arah sudut pandangnya.

```
Procedure Adjust_ViewPort(No:Integer);
Var BMat:Matlx4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
```

Pada Mode ViewPort bukan 0, Camera bergerak mengikuti pesawat dengan bertengger di belakangnya dalam suatu jarak tertentu.

```
    If ViewMode<>0 then
      Begin
```

Jika jarak antara camera dengan pesawat masih lebih kecil dari jarak yang diinginkan maka terus tambah jarak tersebut.

```
        If CtrView<FixViewDistance then
          Begin
            CtrView:=CtrView+20;
            If CtrView>FixViewdistance then;
               CtrView:=FixViewdistance;
          End;
```

Jika jarak antara camera dengan pesawat masih lebih besar dari jarak yang diinginkan maka terus kurangi jarak tersebut.

Dengan teknik ini dihasilkan efek camera yang perlahan-lahan mendekati sasaran seperti yang kita lihat pada game DAYTONA.

```
        If CtrView>FixViewDistance then
          Begin
            CtrView:=CtrView-20;
```

```
        If CtrView<FixViewdistance then;
           CtrView:=FixViewdistance;
      End;
```

Ciptakan sebuah vektor perpindahan yang menunjukkan pergeseran 20 satuan ke sumbu Y (atas) dan CtrView satuan ke arah sumbu Z negative. Ini adalah vektor pergeseran mulai dari titik tengah dari pesawat ke titik baru di belakang agak ke atas dari pesawat. Tujuannya ialah sebagai lokasi dari camera yang melakukan syuting dari belakang pesawat.

```
        BMat[0]:=0;
        BMat[1]:=20;
        BMat[2]:=-CtrView;
        BMat[3]:=1;
```

Tetapi ingat, karena vektor di atas masih relative terhadap koordinat asli, maka perlu dilakukan transformasi dengan menggunakan matrix koordinat frame dari objek, sehingga diperoleh vektor perpindahan yang relative terhadap koordinat frame dari objek.

```
        Mat_Multiply4x1(Box[0].OTMat,BMat);
```

Kemudian lakukan pergeseran dari titik tengah pesawat dengan vektor hasil transformasi tadi untuk mendapatkan koordinat camera.

```
        ViewPort.X:=Box[0].XR+BMat[0];
        ViewPort.Y:=Box[0].YR+BMat[1];
        ViewPort.Z:=Box[0].ZR+BMat[2];
```

Jangan lupa, ambil matrix koordinat frame dan inversenya sebagai matrix koordinat frame dan inverse dari camera, karena sekarang camera memiliki arah koordinat frame yang sama dengan pesawat. (Sudut pandang pesawat sama dengan sudut pandang camera.)

```
        VMat:=Box[0].OTMat;
        IMat:=Box[0].OIMat;
      End;
  End;
```

Pembenahan terakhir dilakukan terhadap main program.

```
Procedure Main;
Var Ch:Char;
    Speed:Integer;
    DZ:Integer;
    DY:Integer;
    DX:Integer;
  Begin
    Ch:='s';
    Speed:=0;
```

Gunakan variabel DZ, DY dan DX sebagai pembatas rotasi sumbu X, Y dan Z dari pesawat. Karena dari matrix koordinat frame dari pesawat tidak dapat diketahui, berapa jauh rotasi sumbu X, Y dan Z telah dilakukan.

```
DZ:=0;
DY:=0;
DX:=0;
```

CtrView sebagai jarak antara camera ke pesawat pertama kali ialah 50 satuan dari pesawat.

```
CtrView:=50;
Repeat
  Repeat
    FillWord(Buff^,32000,0);
    Remove_PolyList;
    ViewPort_Transform;
    Project_Object;
    View_Volume_Filter;
    BackFace_Filter;
    Painter_Sort;
    Display_PList_Shading;
    MoveTo(Buff^,Screen^);
    Case ch of
```

Jika ditekan tombol bawah, lakukan rotasi terhadap sumbu X yang membuat pesawat menengadah ke atas. Batas rotasi hanya dapat dilakukan hingga 30 derajat.

```
#80:Begin
      If DX<30 then
        Begin
          X_Rotation(4);
          Inc(DX,4);
        End;
    End;
```

Lakukan rotasi sumbu X dengan arah minus untuk membuat pesawat menukik ke bawah.

```
#72:Begin
      If DX>-30 then
        Begin
          X_Rotation(-4);
          Dec(DX,4);
        End;
    End;
```

Lakukan rotasi sumbu Z jika ditekan tombol kiri. Pesawat akan berputar berlawanan arah jarum jam. Sehingga sayap kirinya miring ke bawah.

```
#75:Begin
      If DZ>-30 then
        Begin
          Z_Rotation(-4);
          Dec(DZ,4);
        End;
    End;
```

Lakukan rotasi sumbu Z searah jarum jam sehingga sayap kanan pesawat miring ke bawah.

```
#77:Begin
      If DZ<30 then
        Begin
          Z_Rotation(4);
          Inc(DZ,4);
        End;
    End;
'r':X_Rotation(2);
'f':X_Rotation(-2);
```

Gerakkan pesawat maju.

```
'a':Speed:=2;
```

Gerakkan pesawat mundur.

```
'z':Speed:=-2;
```

Kembalikan pesawat ke sudut pandang normal.

```
' ':Begin
      If DX<0 then
        Begin
          If DX>-4 then
            Begin
              X_Rotation(-DX);
              DX:=0;
            End Else
            Begin
              X_Rotation(4);
              Inc(DX,4);
            End;
        End;
      If DX>0 then
        Begin
          If DX<4 then
            Begin
              X_Rotation(-DX);
              DX:=0;
```

```
                End Else
                Begin
                  X_Rotation(-4);
                  Dec(DX,4);
                End;
            End;
          If DZ<0 then
            Begin
              If DZ>-3 then
                Begin
                  Z_Rotation(-DZ);
                  DZ:=0;
                End Else
                Begin
                  Z_Rotation(3);
                  Inc(DZ,3);
                End;
            End;
          If DZ>0 then
            Begin
              If DZ<3 then
                Begin
                  Z_Rotation(-DZ);
                  DZ:=0;
                End Else
                Begin
                  Z_Rotation(-3);
                  Dec(DZ,3);
                End;
            End;
        End;
End;
```

Nah ... secara kenyataan, jika sayap kiri pesawat miring ke bawah maka pesawat akan berotasi terhadap sumbu Y asli (bukan sumbu Y koordinat frame lho) berlawanan arah jarum jam. Teknik ini memang digunakan untuk membelokkan pesawat terbang.

Sehingga kita lakukan rotasi sumbu Y tetapi dengan menggunakan pre-order multiplication karena rotasi yang dilakukan relative terhadap koordinat asli.

```
If DZ<0 then
  Begin
    Y_Rotation2(2);
  End;
```

Jika sayap kanan yang miring ke bawah maka lakukan rotasi sumbu Y koordinat asli searah jarum jam.

```
If DZ>0 then
  Begin
    Y_Rotation2(-2);
  End;
```

Selalu gerakkan pesawat ke arah yang disebutkan pada koordinat framenya.

```
Move_Object(Speed);
```

Selalu update koordinat dan arah dari camera sesuai dengan mode viewport yang dipilih.

```
    Adjust_ViewPort(ViewMode);
  Until Keypressed;
  Ch:=Readkey;
```

Pindah ke mode viewport berikutnya jika ditekan tombol enter.

```
  if Ch=#13 then Inc(ViewMode);
```

Dengan kalkulasi ini maka pada mode viewport 1, 2, 3 dan 4 camera akan memiliki jarak yang semakin jauh, semakin besar nomor mode viewport yang dipilih.

```
    FixViewdistance:=ViewMode*100-50;
    If ViewMode>4 then ViewMode:=0;
  Until Ch=#27;
End;
```

HORE!!! Akhirnya kita berhasil menciptakan sebuah Flight Simulator sederhana. Pemrograman tiga dimensi ini memang kadang agak membuat kepala pusing. Tetapi sebenarnya topik ini sangat menarik untuk dipelajari. Apalagi dalam pembuatan game-game simulator.

Oh ya, pembahasan kita belum selesai sampai di sini lho!!! Lanjutkan ke bab terakhir yaitu tentang optimasi kecepatan.

BAB 9
OPTIMASI KECEPATAN

9.1 Pendahuluan

Wah! Kita telah sampai pada bab terakhir nih. Walaupun ini adalah bab terakhir, tetapi pembahasan kita di sini tidak kalah penting lho dengan bab-bab sebelumnya. Malahan optimasi kecepatan ini adalah merupakan kunci yang menyebabkan suksesnya game-game tiga dimensi di pasaran. Kenapa?

Coba bayangkan, kita mampu menciptakan game dengan teknik tiga dimensi yang handal dan benar-benar realistis seperti terdapat bayangan dari cahaya lampu yang terhalang objek. Juga objek yang ditampilkan sangat halus karena dirakit dari ribuan bahkan ratusan ribu poligon. Di tambah lagi dengan sound system yang mendukung ambient sound dan surround. Pokoknya benar-benar canggih. Tetapi ada satu hal ... game tersebut lambatnya minta ampun karena teknik yang dipergunakan tidak dioptimasi dengan baik. Hasilnya, tidak ada yang mau beli game tersebut karena tidak enak untuk dimainkan.

Kalau kita membicarakan masalah pemrograman tiga dimensi, apa lagi dalam pembuatan game yang bersifat real time renderring (seluruh lingkungan di-render pada saat game berlangsung) bukannya seperti 3D Studio yang di-render terlebih dahulu ke dalam sejumlah gambar, yang kemudian gambar-gambar hasil render-an inilah yang ditampilkan secara cepat.

9.2 Optimasi Flight Simulator

Adakah di antara pembaca yang telah mencoba program kita yang terakhir yaitu B3FLGHT.PAS pada komputer 386 atau 286. Pasti animasi yang dihasilkan lambat. Di komputer Pentium 90 yang saya gunakan saja lumayan lambat lho. Itu pun objek yang ditampilkan di layar monitor masih sekitar 10. Coba kalau jumlah objek mencapai 50, pastilah animasi akan terlihat sangat lambat.

Beberapa hal yang mengakibatkan lambatnya suatu animasi ialah sebagai berikut:

- Penggunaan operasi Real Number yang terlalu sering.
- Penggunaan operasi perkalian, akar, pangkat dan pembagian yang terlalu sering.
- Penggunaan fungsi trigonometri seperti COS dan SIN yang terlalu sering.
- Pemilihan logika yang tidak tepat atau tidak efisien.

Nah, sekarang kita akan menyelidiki bagian mana dari program yang paling sering dipanggil. Yang pertama pada unit VGACAD.PAS. Hmm ... unit ini telah kita optimize dengan penggunaan bahasa assembler untuk merakitnya. Tentu saja kita tidak perlu melakukan optimize lebih lanjut.

Berikutnya ialah pada unit MATH.PAS. Proses perkalian matrix, proses Cosinus dan Sinus yang menggunakan fungsi dari Pascal. Bagian-bagian inilah yang akan kita optimize.

Pertama-tama, pada perkalian matrix. Mengingat kebanyakan isi dari matrix transformasi kita ialah 0 maka pada element yang berharga 0, proses perkalian tidak perlu dilakukan, karena hasil kali pastilah 0.

Kedua, pada fungsi Cosinus dan Sinus. Cara menghindari penggunaan fungsi ini ialah menggunakan Look Up Table atau Hash Table seperti yang kita gunakan pada pembuatan efek transparant. Kita ciptakan dua buah array yang berisi harga Sinus dan Cosinus dari index-nya. Dengan demikian untuk mencari Cos 45 dilakukan proses pembacaan array index 45 karena isinya menunjukkan Cos 45. Mudah kan?

Ingat, sudut yang kita gunakan tidak hanya bilangan bulat, bisa terjadi suatu saat kita harus melakukan Cos 43,2 misalnya. Karena index dari array selalu bilangan bulat, maka untuk pemecahannya, kita ciptakan Array dengan sudut * 100 sebagai indexnya. Untuk itu array kita akan sebesar 360 x 100 = 36.000 element. Dengan cara ini maka Cos 43,2 sama dengan Table Cos[43,2*100] = Cos[4320].

Untuk menghemat penggunaan memori, maka cukup dilakukan penyimpanan hasil Cos dan Sin antara 0 hingga 90 derajat (dalam hal ini ialah index 0 hingga 9000), mengingat kita mengetahui secara geometri bahwa hasil Cos dan Sin dari sudut yang lebih besar dari 90 derajat dapat dicari dengan menggunakan system quadrant.

Untuk Sudut a = 0-90	Cos a = Cos a
Untuk Sudut a = 90-180	Cos a =-Cos (180-a)
Untuk Sudut a = 180-270	Cos a =-Cos (a-180)
Untuk Sudut a = 270-360	Cos a = Cos (360-a)
Untuk Sudut a = 0-90	Sin a = Sin a
Untuk Sudut a = 90-180	Sin a = Sin (180-a)
Untuk Sudut a = 180-270	Sin a =-Sin (a-180)
Untuk Sudut a = 270-360	Sin a =-Sin (360-a)

Dengan cara ini kita cukup memerlukan informasi Cos dan Sin antara sudut 0 hingga 90 derajat untuk mencari Cos dan Sin antara sudut 0 hingga 360 derajat.

Ok, inilah hasil modifikasi dari unit MATH.PAS.

```
{-----------------------------------------------}
{ Program     : MATH3.PAS                       }
{ Coder       : Agustinus Nalwan                }
{ Description: Unit Matrix dan Vektor yang telah}
{              di-optimize.                     }
{-----------------------------------------------}

Unit Math3;

Interface

Uses Crt,Dos;

Const dtr = pi/180;
      rtd = 180/pi;

Type Mat4x4 = array[0..3,0..3] of Real;
     Mat1x4 = array[0..3] of Real;
     Vec3D  = array[0..2] of Real;
     Table  = array[0..9000] of Real;

Var TSin : ^Table;
    TCos : ^Table;
    A    : Longint;

Function DToR(Deg:Real) : Real;
Function RToD(Rad:Real) : Real;
Function Cosinus(Deg:Longint) : Real;
Function Sinus(Deg:Longint)   : Real;
Function Tangent(Deg:Longint) : Real;
Function arCos(Num:Real) : Real;
Function arSin(Num:Real) : Real;
```

```
Function arTan(Num:Real) : Real;

Procedure Mat_Identity(Var A:Mat4x4);
Procedure Mat_XRotate(Var A:Mat4x4;Deg:Longint);
Procedure Mat_YRotate(Var A:Mat4x4;Deg:Longint);
Procedure Mat_ZRotate(Var A:Mat4x4;Deg:Longint);
Procedure Mat_Dilatasi(Var A:Mat4x4;tx,ty,tz:Real);
Procedure Mat_Scaling(Var A:Mat4x4;sx,sy,sz:Real);
Procedure Mat_Multiply4x4(Var A:Mat4x4;B:Mat4x4);
Procedure Mat_Multiply4x1(A:Mat4x4;Var B:Mat1x4);
Procedure Mat_Add(Var A:Mat4x4;B:Mat4x4);
Procedure Mat_Subtract(Var A:Mat4x4;B:Mat4x4);

Procedure Vect_Add(Var A:Vec3D;B:Vec3D);
Procedure Vect_Sub(Var A:Vec3D;B:Vec3D);
Procedure Vect_Unit(Var A:Vec3D;B:Vec3D);
Function  Vect_Mag(A:Vec3D) : Real;
Function  Vect_Dot(A,B:Vec3D) : Real;
Function  Vect_Angle(A,B:Vec3D) : Real;
Procedure Vect_Cross(Var A:Vec3D;B,C:Vec3D);

Implementation

Function DToR(Deg:Real) : Real;
  Begin
    DToR:=Deg*dtr;
  End;

Function RToD(Rad:Real) : Real;
  Begin
    RToD:=Rad*rtd;
  End;

Function Cosinus(Deg:Longint) : Real;
  Begin
    If Deg>36000 then Deg:=Deg-36000;
    If Deg<0 then Deg:=36000+Deg;
    If (Deg<=9000) and (Deg>=0) then
      Begin
        Cosinus:=TCos^[Deg];
      End Else
    If (Deg>=9000) and (Deg<=18000) then
      Begin
        Cosinus:=-TCos^[18000-Deg];
      End Else
    If (Deg>=18000) and (Deg<=27000) then
      Begin
        Cosinus:=-(TCos^[Deg-18000]);
      End Else
        Cosinus:=TCos^[36000-Deg];
  End;

Function Sinus(Deg:Longint)    : Real;
  Begin
    If Deg>36000 then Deg:=Deg-36000;
    If Deg<0 then Deg:=36000+Deg;
    If (Deg<=9000) and (Deg>=0) then
      Begin
        Sinus:=TSin^[Deg];
      End Else
    If (Deg>=9000) and (Deg<=18000) then
```

```
      Begin
        Sinus:=TSin^[18000-Deg];
      End Else
    If (Deg>=18000) and (Deg<=27000) then
      Begin
        Sinus:=-(TSin^[Deg-18000]);
      End Else
        Sinus:=-TSin^[36000-Deg];
  End;

Function Tangent(Deg:Longint)    : Real;
  Begin
    Tangent:=Sinus(Deg)/Cosinus(Deg);
  End;

Function arCos(Num:Real) : Real;
  Begin
    If Num=0 then arCos:=pi/2 else
      arCos:=ArcTan (sqrt (1-sqr (Num)) /Num);
  End;

Function arSin(Num:Real) : Real;
  Begin
    If Num=1 then arSin:=pi/2 else
      arSin:=ArcTan (Num/sqrt (1-sqr (Num)));
  End;

Function arTan(Num:Real) : Real;
  Begin
    arTan:=ArcTan(Num);
  End;

Procedure Mat_Identity(Var A:Mat4x4);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_XRotate(Var A:Mat4x4;Deg:Longint);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=Cosinus(Deg);A[1,2]:=Sinus(Deg);A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=-Sinus(Deg);A[2,2]:=Cosinus(Deg);A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_YRotate(Var A:Mat4x4;Deg:Longint);
  Begin
    A[0,0]:=Cosinus(Deg);A[0,1]:=0;A[0,2]:=-Sinus(Deg);A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=Sinus(Deg);A[2,1]:=0;A[2,2]:=Cosinus(Deg);A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_ZRotate(Var A:Mat4x4;Deg:Longint);
  Begin
```

```
    A[0,0]:=Cosinus(Deg);A[0,1]:=Sinus(Deg);A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=-Sinus(Deg);A[1,1]:=Cosinus(Deg);A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_Dilatasi(Var A:Mat4x4;tx,ty,tz:Real);
  Begin
    A[0,0]:=1;A[0,1]:=0;A[0,2]:=0;A[0,3]:=tx;
    A[1,0]:=0;A[1,1]:=1;A[1,2]:=0;A[1,3]:=ty;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=1;A[2,3]:=tz;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_Scaling(Var A:Mat4x4;sx,sy,sz:Real);
  Begin
    A[0,0]:=sx;A[0,1]:=0;A[0,2]:=0;A[0,3]:=0;
    A[1,0]:=0;A[1,1]:=sy;A[1,2]:=0;A[1,3]:=0;
    A[2,0]:=0;A[2,1]:=0;A[2,2]:=sz;A[2,3]:=0;
    A[3,0]:=0;A[3,1]:=0;A[3,2]:=0;A[3,3]:=1;
  End;

Procedure Mat_Multiply4x4(Var A:Mat4x4;B:Mat4x4);
Var d,e,c:Integer;
    BuffMat:Mat4x4;
    Bf:Real;
  Begin
    For d:=0 to 3 do
      For e:=0 to 3 do
        Begin
          BuffMat[e,d]:=0;
          For c:=0 to 3 do
            Begin
              If (A[e,c]=0) or (B[c,d]=0) then else
                Begin
                  Bf:=A[e,c]*B[c,d];
                  BuffMat[e,d]:=BuffMat[e,d]+Bf;
                End;
            End;
        End;
    A:=BuffMat;
  End;

Procedure Mat_Multiply4x1(A:Mat4x4;Var B:Mat1x4);
Var e,c:Integer;
    BuffMat:Mat1x4;
  Begin
    For e:=0 to 3 do
      Begin
        BuffMat[e]:=0;
        For c:=0 to 3 do
          BuffMat[e]:=BuffMat[e]+A[e,c]*B[c];
      End;
    B:=BuffMat;
  End;

Procedure Mat_Add(Var A:Mat4x4;B:Mat4x4);
Var c,d:Integer;
  Begin
```

```
    For c:=0 to 3 do
      For d:=0 to 3 do
        A[c,d]:=A[c,d]+B[c,d];
  End;

Procedure Mat_Subtract(Var A:Mat4x4;B:Mat4x4);
Var c,d:Integer;
  Begin
    For c:=0 to 3 do
      For d:=0 to 3 do
        A[c,d]:=A[c,d]-B[c,d];
  End;

Procedure Vect_Add(Var A:Vec3D;B:Vec3D);
Var c:Integer;
  Begin
    For c:=0 to 2 do
      A[c]:=A[c]+B[c];
  End;

Procedure Vect_Sub(Var A:Vec3D;B:Vec3D);
Var c:Integer;
  Begin
    For c:=0 to 2 do
      A[c]:=A[c]-B[c];
  End;

Function Vect_Mag(A:Vec3D) : Real;
  Begin
    Vect_Mag:=Sqrt(A[0]*A[0]+A[1]*A[1]+A[2]*A[2]);
  End;

Procedure Vect_Unit(Var A:Vec3D;B:Vec3D);
Var M:Real;
  Begin
    M:=Vect_Mag(B);
    A[0]:=B[0]/M;
    A[1]:=B[1]/M;
    A[2]:=B[2]/M;
  End;

Function  Vect_Dot(A,B:Vec3D) : Real;
  Begin
    Vect_Dot:=A[0]*B[0]+A[1]*B[1]+A[2]*B[2];
  End;

Function  Vect_Angle(A,B:Vec3D) : Real;
  Begin

   Vect_Angle:=RToD(arCos(Vect_Dot(A,B)/(Vect_Mag(A)*Vect_Mag(B))));
  End;

Procedure Vect_Cross(Var A:Vec3D;B,C:Vec3D);
  Begin
    A[0]:=B[1]*C[2]-B[2]*C[1];
    A[1]:=-B[0]*C[2]+B[2]*C[0];
    A[2]:=B[0]*C[1]-B[1]*C[0];
  End;
```

```
Begin
  GetMem(TSin,SizeOf(Table));
  GetMem(TCos,SizeOf(Table));
  For A:=0 to 9000 do
    Begin
      TSin^[A]:=Sin((A/100)*pi/180);
      TCos^[A]:=Cos((A/100)*pi/180);
    End;
End.
```

PENJELASAN LISTING PROGRAM

Sekarang marilah kita lihat bagian-bagian yang dimodifikasi.

Pada bagian data structure, ditambahkan table Sinus dan Cosinus yang berindex 0 hingga 9000. Ini mewakili sudut antara 0 hingga 90 derajat.

```
Type Mat4x4 = array[0..3,0..3] of Real;
     Mat1x4 = array[0..3] of Real;
     Vec3D  = array[0..2] of Real;
     Table  = array[0..9000] of Real;

Var TSin : ^Table;
    TCos : ^Table;
    A    : Longint;

Function DToR(Deg:Real) : Real;
  Begin
```

Konversi dari degree ke radiant cukup membutuhkan sebuah perkalian di samping cara kita yang terdahulu yaitu Deg * pi / 180.

Di sini pi / 180 telah dihitung dan disimpan dalam variabel dtr.

```
    DToR:=Deg*dtr;
  End;

Function RToD(Rad:Real) : Real;
  Begin
```

Konversi dari degree ke radiant cukup membutuhkan sebuah perkalian di samping cara kita yang terdahulu yaitu Deg * 180 / pi.

Di sini 180/pi telah dihitung dan disimpan dalam variabel rtd.

```
    RToD:=Rad*rtd;
  End;

Function Cosinus(Deg:Longint) : Real;
  Begin
```

Dilakukan pengecekan apakah sudut melebihi 360 derajat. Jika ya, kembalikan sudut sehingga berkisar antara 0 hingga 360 derajat.

```
    If Deg>36000 then Deg:=Deg-36000;
    If Deg<0 then Deg:=36000+Deg;
```

Jika sudut pada quadrant I yaitu antara 0-90 derajat maka langsung lakukan look up.

```
    If (Deg<=9000) and (Deg>=0) then
      Begin
        Cosinus:=TCos^[Deg];
      End Else
```

Untuk quadrant II, lakukan lookup terhadap 180-Sudut.

```
    If (Deg>=9000) and (Deg<=18000) then
      Begin
        Cosinus:=-TCos^[18000-Deg];
      End Else
```

Untuk quadrant III, lakukan lookup terhadap Sudut-180.

```
    If (Deg>=18000) and (Deg<=27000) then
      Begin
        Cosinus:=-(TCos^[Deg-18000]);
      End Else
```

Untuk quadrant IV, lakukan lookup terhadap 360-Sudut.

```
        Cosinus:=TCos^[36000-Deg];
  End;

Function Sinus(Deg:Longint)    : Real;
  Begin
```

Lakukan proses yang sama seperti pada proses Cosinus.

```
    If Deg>36000 then Deg:=Deg-36000;
    If Deg<0 then Deg:=36000+Deg;
    If (Deg<=9000) and (Deg>=0) then
      Begin
        Sinus:=TSin^[Deg];
      End Else
    If (Deg>=9000) and (Deg<=18000) then
      Begin
        Sinus:=TSin^[18000-Deg];
      End Else
    If (Deg>=18000) and (Deg<=27000) then
      Begin
        Sinus:=-(TSin^[Deg-18000]);
      End Else
        Sinus:=-TSin^[36000-Deg];
  End;
```

```
Procedure Mat_Multiply4x4(Var A:Mat4x4;B:Mat4x4);
Var d,e,c:Integer;
    BuffMat:Mat4x4;
    Bf:Real;
  Begin
    For d:=0 to 3 do
      For e:=0 to 3 do
        Begin
          BuffMat[e,d]:=0;
          For c:=0 to 3 do
            Begin
```

Test apakah salah satu dari element yang hendak dikalikan 0 atau tidak. Jika ya maka tidak perlu dilakukan perkalian dan penambahan pada BuffMat[e,d].

Dengan ini maka kita dapat menghindari perkalian untuk element yang memiliki harga 0. Sehingga jumlah perkalian akan berkurang.

```
              If (A[e,c]=0) or (B[c,d]=0) then else
                Begin
                  Bf:=A[e,c]*B[c,d];
                  BuffMat[e,d]:=BuffMat[e,d]+Bf;
                End;
            End;
        End;
    A:=BuffMat;
  End;

Begin
```

Pesan memori untuk tabel Sinus dan Cosinus.

```
  GetMem(TSin,SizeOf(Table));
  GetMem(TCos,SizeOf(Table));
```

Hitung hasil dari Sinus dan Cosinus dan masukkan ke dalam look up table.

```
  For A:=0 to 9000 do
    Begin
      TSin^[A]:=Sin((A/100)*pi/180);
      TCos^[A]:=Cos((A/100)*pi/180);
    End;
End.
```

Selanjutnya marilah kita lakukan optimasi terhadap program B3FLGHT.PAS kita yang lalu sehingga mampu menampilkan lebih banyak objek dengan kecepatan yang memadai.

Seperti yang telah kita lakukan pada unit MATH.PAS, kita lakukan analisa

terhadap program B3FLGHT.PAS ini.

Proses tersering selain proses yang bersangkutan dengan VGACAD.PAS ialah proses transformasi yang melibatkan perkalian matrix dan pengisian matrix dengan fungsi COS dan SIN. Sebagai pemecahan, cukup dilakukan penggunaan unit MATH3.PAS yang merupakan unit yang telah dioptimasi. Setelah kita menggunakan unit MATH3.PAS ini, kecepatan animasi terlihat naik drastis sekitar tiga kali lebih cepat dari program B3FLGHT.PAS yang menggunakan MATH.PAS yang belum dioptimasi. Hal ini membuktikan proses perkalian telah jauh berkurang. Juga Sin dan Cos dari look up table berperanan penting pada optimasi kecepatan.

Nah, karena seluruh proses seperti filter, polygon shading, painter sort dan proyeksi dilakukan terhadap seluruh poligon dalam polygon list maka target optimasi berikutnya ialah mengurangi jumlah poligon dalam polygon list semaksimal mungkin.

Dalam hal ini telah dilakukan beberapa filter untuk mengurangi poligon dalam polygon list. Tetapi filter-filter ini dilakukan setelah dilakukannya proses Inverse Transform dan Object Projection. Dengan kata lain, seluruh poligon akan mengalami kedua proses tersebut. Hmmm ... jadi inilah yang menjadi penyebab lambatnya animasi kita. Lantas bagaimana pemecahannya. Kita ciptakan sebuah filter yang akan dilakukan sebelum proses Object Projection. Pembuatan filter sebelum proses Inverse Transform hampir tidak mungkin karena kita belum mengetahui koordinat semu dari objek sebelum dilakukannya proses tersebut.

Filter yang akan dilakukan setelah Inverse Transform dan sebelum Object Projection ialah View Volume Filter untuk objek. Di sini dilihat apakah koordinat dari objek berada di luar View Volume. Jika ya, maka tidak perlu dilakukan projection dan seluruh poligon dari objek tidak perlu dimasukkan ke dalam Polygon List.

Memang cara ini mengandung resiko, karena dengan mendeteksi koordinat tengah dari objek saja tidak cukup untuk mengetahui apakah objek berada di dalam atau di luar view volume, karena itu pun juga tergantung dari ukuran objek. Tetapi dalam hal ini, karena objek yang kita gunakan tidak terlalu besar, maka resiko kita ambil!!!

Berikut ini adalah hasil optimasi yang telah kita lakukan.

```
{-----------------------------------------------                }
{ Program     : B3OPTMZ.PAS                                    }
{ Coder       : Agustinus Nalwan                               }
{ Description: Program Flight Simulator yang telah             }
{              di-optimize.                                    }
{-----------------------------------------------                }

Uses Math3,Crt,Dos,VgaCad;

Const View_Distance = 400;
      Total_Obj     = 50;

Type
     Poly_Rec = Record
       Vertex      : Array[0..4] of Integer;
       TotalVertex : Integer;
       Color       : Byte;
     End;

     Vertex_Rec = Record
       XR, YR, ZR : Real;
       X, Y, Z    : Real;
       Xp,Yp      : Longint;
     End;

     ObjRect = Record
       Color      : Byte;
       VertexBank : Array[0..80] of Vertex_Rec;
       Polygon    : Array[0..50] of Poly_Rec;
       TotalVertex: Integer;
       TotalPoly  : Integer;
       X,Y,Z      : Real;
       XR,YR,ZR   : Real;
       OTMat      : Mat4x4;
       OIMat      : Mat4x4;
     End;

     PolyList_Rec = Record
       ObjNo,PolyNo : Integer;
       Active       : Boolean;
       ZMin         : Real;
       ZMax         : Real;
       XMin         : Real;
       XMax         : Real;
       YMin         : Real;
       YMax         : Real;
       Color        : Integer;
     End;

     ViewPort_Rec = Record
       X, Y, Z  : Real;
       Dx,Dy,Dz : Real;
     End;

Var A,B:Integer;
    Box:array[0..Total_Obj] of ^ObjRect;
    PolygonList:array[0..1000] of ^PolyList_Rec;
    ViewPort:ViewPort_Rec;
```

```
    ListCtr:Integer;
    buff:Pointer;
    VMat:Mat4x4;
    IMat:Mat4x4;
    ViewMode:Integer;
    CtrView,FixViewDistance:Longint;

Procedure Pll_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round(X);pY:=100-Round(Y);
  End;

Procedure Psv_Projection(X,Y,Z:Real;Var pX,pY:Longint);
  Begin
    pX:=160+Round((View_Distance*X)/Z);
    pY:=100-Round((View_Distance*Y)/Z);
  End;

Procedure Create_Box(No:Integer;X,Y,Z,lx,ly,lz:Real;Col:Byte);
  Begin
    Box[No]^.VertexBank[0].XR:=-lx;Box[No]^.VertexBank[0].YR:=ly;
        Box[No]^.VertexBank[0].ZR:=-lz;
    Box[No]^.VertexBank[1].XR:=lx;Box[No]^.VertexBank[1].YR:=ly;
        Box[No]^.VertexBank[1].ZR:=-lz;
    Box[No]^.VertexBank[2].XR:=lx;Box[No]^.VertexBank[2].YR:=-ly;
        Box[No]^.VertexBank[2].ZR:=-lz;
    Box[No]^.VertexBank[3].XR:=-lx;Box[No]^.VertexBank[3].YR:=-ly;
        Box[No]^.VertexBank[3].ZR:=-lz;
    Box[No]^.VertexBank[4].XR:=-lx;Box[No]^.VertexBank[4].YR:=ly;
        Box[No]^.VertexBank[4].ZR:=lz;
    Box[No]^.VertexBank[5].XR:=lx;Box[No]^.VertexBank[5].YR:=ly;
        Box[No]^.VertexBank[5].ZR:=lz;
    Box[No]^.VertexBank[6].XR:=lx;Box[No]^.VertexBank[6].YR:=-ly;
        Box[No]^.VertexBank[6].ZR:=lz;
    Box[No]^.VertexBank[7].XR:=-lx;Box[No]^.VertexBank[7].YR:=-ly;
        Box[No]^.VertexBank[7].ZR:=lz;
    Box[No]^.Polygon[0].TotalVertex:=4;
    Box[No]^.Polygon[1].TotalVertex:=4;
    Box[No]^.Polygon[2].TotalVertex:=4;
    Box[No]^.Polygon[3].TotalVertex:=4;
    Box[No]^.Polygon[4].TotalVertex:=4;
    Box[No]^.Polygon[5].TotalVertex:=4;

    Box[No]^.Polygon[0].Vertex[0]:=0;
    Box[No]^.Polygon[0].Vertex[1]:=3;
    Box[No]^.Polygon[0].Vertex[2]:=2;
    Box[No]^.Polygon[0].Vertex[3]:=1;

    Box[No]^.Polygon[1].Vertex[0]:=0;
    Box[No]^.Polygon[1].Vertex[1]:=1;
    Box[No]^.Polygon[1].Vertex[2]:=5;
    Box[No]^.Polygon[1].Vertex[3]:=4;

    Box[No]^.Polygon[2].Vertex[0]:=4;
    Box[No]^.Polygon[2].Vertex[1]:=5;
    Box[No]^.Polygon[2].Vertex[2]:=6;
    Box[No]^.Polygon[2].Vertex[3]:=7;
```

```
    Box[No]^.Polygon[3].Vertex[0]:=7;
    Box[No]^.Polygon[3].Vertex[1]:=6;
    Box[No]^.Polygon[3].Vertex[2]:=2;
    Box[No]^.Polygon[3].Vertex[3]:=3;

    Box[No]^.Polygon[4].Vertex[0]:=4;
    Box[No]^.Polygon[4].Vertex[1]:=7;
    Box[No]^.Polygon[4].Vertex[2]:=3;
    Box[No]^.Polygon[4].Vertex[3]:=0;

    Box[No]^.Polygon[5].Vertex[0]:=5;
    Box[No]^.Polygon[5].Vertex[1]:=1;
    Box[No]^.Polygon[5].Vertex[2]:=2;
    Box[No]^.Polygon[5].Vertex[3]:=6;

    Box[No]^.Polygon[0].Color:=Col;
    Box[No]^.Polygon[1].Color:=Col;
    Box[No]^.Polygon[2].Color:=Col;
    Box[No]^.Polygon[3].Color:=Col;
    Box[No]^.Polygon[4].Color:=Col;
    Box[No]^.Polygon[5].Color:=Col;

    Box[No]^.TotalPoly:=6;
    Box[No]^.TotalVertex:=8;
    Box[No]^.XR:=X;Box[No]^.YR:=Y;Box[No]^.ZR:=Z;
  End;

Procedure SVB(ObjNo,VxNo:Integer;X,Y,Z:Real);
  Begin
    Box[ObjNo]^.VertexBank[VxNo].XR:=X;
    Box[ObjNo]^.VertexBank[VxNo].YR:=Y;
    Box[ObjNo]^.VertexBank[VxNo].ZR:=Z;
  End;

Procedure Create_Ship(No:Integer;X,Y,Z:Real);
Var Bf:Integer;
  Begin
    SVB(No,0,-100,3,0);
    SVB(No,1,-100,3,-10);
    SVB(No,2,-100,2,-10);
    SVB(No,3,-100,2,0);

    SVB(No,4,-60,2,5);
    SVB(No,5,-60,2,-10);
    SVB(No,6,-60,-2,-10);
    SVB(No,7,-60,-2,5);

    SVB(No,8,-40,5,13);
    SVB(No,9,-40,5,-10);
    SVB(No,10,-40,-2,-10);
    SVB(No,11,-40,-2,13);

    SVB(No,12,-40,9,55);
    SVB(No,13,-15,9,55);
    SVB(No,14,-15,-7,45);
    SVB(No,15,-40,-7,45);
```

```
SVB(No,16,-40,9,-10);
SVB(No,17,-20,9,-10);
SVB(No,18,-20,-7,-15);
SVB(No,19,-40,-7,-15);

SVB(No,20,-30,25,-8);
SVB(No,21,-30,25,-10);
SVB(No,22,-31,9,-10);
SVB(No,23,-31,9,0);
SVB(No,24,-29,9,0);
SVB(No,25,-29,9,-10);

SVB(No,52,-18,0,35);
SVB(No,53,18,0,35);
SVB(No,54,-10,15,10);
SVB(No,55,10,15,10);

SVB(No,56,-10,5,0);
SVB(No,57,10,5,0);
SVB(No,58,-18,0,0);
SVB(No,59,18,0,0);

For A:=0 to 25 do
  Begin
    Box[No]^.VertexBank[26+A]:=Box[No]^.VertexBank[A];
    Box[No]^.VertexBank[26+A].XR:=-Box[No]^.VertexBank[26+A].XR;
  End;

For A:=0 to 60 do
  Begin
    Box[No]^.Polygon[A].TotalVertex:=4;
  End;

Box[No]^.Polygon[18].TotalVertex:=3;
Box[No]^.Polygon[19].TotalVertex:=3;
Box[No]^.Polygon[38].TotalVertex:=3;
Box[No]^.Polygon[39].TotalVertex:=3;

Box[No]^.Polygon[0].Vertex[0]:=7;
Box[No]^.Polygon[0].Vertex[1]:=11;
Box[No]^.Polygon[0].Vertex[2]:=10;
Box[No]^.Polygon[0].Vertex[3]:=6;

Box[No]^.Polygon[1].Vertex[0]:=0;
Box[No]^.Polygon[1].Vertex[1]:=3;
Box[No]^.Polygon[1].Vertex[2]:=2;
Box[No]^.Polygon[1].Vertex[3]:=1;

Box[No]^.Polygon[2].Vertex[0]:=0;
Box[No]^.Polygon[2].Vertex[1]:=1;
Box[No]^.Polygon[2].Vertex[2]:=5;
Box[No]^.Polygon[2].Vertex[3]:=4;

Box[No]^.Polygon[3].Vertex[0]:=4;
Box[No]^.Polygon[3].Vertex[1]:=5;
Box[No]^.Polygon[3].Vertex[2]:=9;
Box[No]^.Polygon[3].Vertex[3]:=8;
```

```
Box[No]^.Polygon[4].Vertex[0]:=1;
Box[No]^.Polygon[4].Vertex[1]:=2;
Box[No]^.Polygon[4].Vertex[2]:=6;
Box[No]^.Polygon[4].Vertex[3]:=5;

Box[No]^.Polygon[5].Vertex[0]:=5;
Box[No]^.Polygon[5].Vertex[1]:=6;
Box[No]^.Polygon[5].Vertex[2]:=10;
Box[No]^.Polygon[5].Vertex[3]:=9;

Box[No]^.Polygon[6].Vertex[0]:=8;
Box[No]^.Polygon[6].Vertex[1]:=9;
Box[No]^.Polygon[6].Vertex[2]:=10;
Box[No]^.Polygon[6].Vertex[3]:=11;

Box[No]^.Polygon[7].Vertex[0]:=0;
Box[No]^.Polygon[7].Vertex[1]:=4;
Box[No]^.Polygon[7].Vertex[2]:=7;
Box[No]^.Polygon[7].Vertex[3]:=3;

Box[No]^.Polygon[8].Vertex[0]:=4;
Box[No]^.Polygon[8].Vertex[1]:=8;
Box[No]^.Polygon[8].Vertex[2]:=11;
Box[No]^.Polygon[8].Vertex[3]:=7;

Box[No]^.Polygon[9].Vertex[0]:=3;
Box[No]^.Polygon[9].Vertex[1]:=7;
Box[No]^.Polygon[9].Vertex[2]:=6;
Box[No]^.Polygon[9].Vertex[3]:=2;

Box[No]^.Polygon[10].Vertex[0]:=12;
Box[No]^.Polygon[10].Vertex[1]:=15;
Box[No]^.Polygon[10].Vertex[2]:=19;
Box[No]^.Polygon[10].Vertex[3]:=16;

Box[No]^.Polygon[11].Vertex[0]:=12;
Box[No]^.Polygon[11].Vertex[1]:=16;
Box[No]^.Polygon[11].Vertex[2]:=17;
Box[No]^.Polygon[11].Vertex[3]:=13;

Box[No]^.Polygon[12].Vertex[0]:=12;
Box[No]^.Polygon[12].Vertex[1]:=13;
Box[No]^.Polygon[12].Vertex[2]:=14;
Box[No]^.Polygon[12].Vertex[3]:=15;

Box[No]^.Polygon[13].Vertex[0]:=13;
Box[No]^.Polygon[13].Vertex[1]:=17;
Box[No]^.Polygon[13].Vertex[2]:=18;
Box[No]^.Polygon[13].Vertex[3]:=14;

Box[No]^.Polygon[14].Vertex[0]:=15;
Box[No]^.Polygon[14].Vertex[1]:=14;
Box[No]^.Polygon[14].Vertex[2]:=18;
Box[No]^.Polygon[14].Vertex[3]:=19;

Box[No]^.Polygon[15].Vertex[0]:=16;
Box[No]^.Polygon[15].Vertex[1]:=19;
Box[No]^.Polygon[15].Vertex[2]:=18;
```

```
Box[No]^.Polygon[15].Vertex[3]:=17;

Box[No]^.Polygon[16].Vertex[0]:=20;
Box[No]^.Polygon[16].Vertex[1]:=23;
Box[No]^.Polygon[16].Vertex[2]:=22;
Box[No]^.Polygon[16].Vertex[3]:=21;

Box[No]^.Polygon[17].Vertex[0]:=20;
Box[No]^.Polygon[17].Vertex[1]:=21;
Box[No]^.Polygon[17].Vertex[2]:=25;
Box[No]^.Polygon[17].Vertex[3]:=24;

Box[No]^.Polygon[18].Vertex[0]:=20;
Box[No]^.Polygon[18].Vertex[1]:=24;
Box[No]^.Polygon[18].Vertex[2]:=23;

Box[No]^.Polygon[19].Vertex[0]:=21;
Box[No]^.Polygon[19].Vertex[1]:=22;
Box[No]^.Polygon[19].Vertex[2]:=25;

Box[No]^.Polygon[20].Vertex[0]:=7;
Box[No]^.Polygon[20].Vertex[1]:=11;
Box[No]^.Polygon[20].Vertex[2]:=10;
Box[No]^.Polygon[20].Vertex[3]:=6;

Box[No]^.Polygon[21].Vertex[0]:=0;
Box[No]^.Polygon[21].Vertex[1]:=3;
Box[No]^.Polygon[21].Vertex[2]:=2;
Box[No]^.Polygon[21].Vertex[3]:=1;

Box[No]^.Polygon[22].Vertex[0]:=0;
Box[No]^.Polygon[22].Vertex[1]:=1;
Box[No]^.Polygon[22].Vertex[2]:=5;
Box[No]^.Polygon[22].Vertex[3]:=4;

Box[No]^.Polygon[23].Vertex[0]:=4;
Box[No]^.Polygon[23].Vertex[1]:=5;
Box[No]^.Polygon[23].Vertex[2]:=9;
Box[No]^.Polygon[23].Vertex[3]:=8;

Box[No]^.Polygon[24].Vertex[0]:=1;
Box[No]^.Polygon[24].Vertex[1]:=2;
Box[No]^.Polygon[24].Vertex[2]:=6;
Box[No]^.Polygon[24].Vertex[3]:=5;

Box[No]^.Polygon[25].Vertex[0]:=5;
Box[No]^.Polygon[25].Vertex[1]:=6;
Box[No]^.Polygon[25].Vertex[2]:=10;
Box[No]^.Polygon[25].Vertex[3]:=9;

Box[No]^.Polygon[26].Vertex[0]:=8;
Box[No]^.Polygon[26].Vertex[1]:=9;
Box[No]^.Polygon[26].Vertex[2]:=10;
Box[No]^.Polygon[26].Vertex[3]:=11;

Box[No]^.Polygon[27].Vertex[0]:=0;
Box[No]^.Polygon[27].Vertex[1]:=4;
```

```
Box[No]^.Polygon[27].Vertex[2]:=7;
Box[No]^.Polygon[27].Vertex[3]:=3;

Box[No]^.Polygon[28].Vertex[0]:=4;
Box[No]^.Polygon[28].Vertex[1]:=8;
Box[No]^.Polygon[28].Vertex[2]:=11;
Box[No]^.Polygon[28].Vertex[3]:=7;

Box[No]^.Polygon[29].Vertex[0]:=3;
Box[No]^.Polygon[29].Vertex[1]:=7;
Box[No]^.Polygon[29].Vertex[2]:=6;
Box[No]^.Polygon[29].Vertex[3]:=2;

Box[No]^.Polygon[30].Vertex[0]:=12;
Box[No]^.Polygon[30].Vertex[1]:=15;
Box[No]^.Polygon[30].Vertex[2]:=19;
Box[No]^.Polygon[30].Vertex[3]:=16;

Box[No]^.Polygon[31].Vertex[0]:=12;
Box[No]^.Polygon[31].Vertex[1]:=16;
Box[No]^.Polygon[31].Vertex[2]:=17;
Box[No]^.Polygon[31].Vertex[3]:=13;

Box[No]^.Polygon[32].Vertex[0]:=12;
Box[No]^.Polygon[32].Vertex[1]:=13;
Box[No]^.Polygon[32].Vertex[2]:=14;
Box[No]^.Polygon[32].Vertex[3]:=15;

Box[No]^.Polygon[33].Vertex[0]:=13;
Box[No]^.Polygon[33].Vertex[1]:=17;
Box[No]^.Polygon[33].Vertex[2]:=18;
Box[No]^.Polygon[33].Vertex[3]:=14;

Box[No]^.Polygon[34].Vertex[0]:=15;
Box[No]^.Polygon[34].Vertex[1]:=14;
Box[No]^.Polygon[34].Vertex[2]:=18;
Box[No]^.Polygon[34].Vertex[3]:=19;

Box[No]^.Polygon[35].Vertex[0]:=16;
Box[No]^.Polygon[35].Vertex[1]:=19;
Box[No]^.Polygon[35].Vertex[2]:=18;
Box[No]^.Polygon[35].Vertex[3]:=17;

Box[No]^.Polygon[36].Vertex[0]:=20;
Box[No]^.Polygon[36].Vertex[1]:=23;
Box[No]^.Polygon[36].Vertex[2]:=22;
Box[No]^.Polygon[36].Vertex[3]:=21;

Box[No]^.Polygon[37].Vertex[0]:=20;
Box[No]^.Polygon[37].Vertex[1]:=21;
Box[No]^.Polygon[37].Vertex[2]:=25;
Box[No]^.Polygon[37].Vertex[3]:=24;

Box[No]^.Polygon[38].Vertex[0]:=20;
Box[No]^.Polygon[38].Vertex[1]:=24;
Box[No]^.Polygon[38].Vertex[2]:=23;
```

```
Box[No]^.Polygon[39].Vertex[0]:=21;
Box[No]^.Polygon[39].Vertex[1]:=22;
Box[No]^.Polygon[39].Vertex[2]:=25;

Box[No]^.Polygon[40].Vertex[0]:=52;
Box[No]^.Polygon[40].Vertex[1]:=54;
Box[No]^.Polygon[40].Vertex[2]:=55;
Box[No]^.Polygon[40].Vertex[3]:=53;

Box[No]^.Polygon[41].Vertex[0]:=54;
Box[No]^.Polygon[41].Vertex[1]:=56;
Box[No]^.Polygon[41].Vertex[2]:=57;
Box[No]^.Polygon[41].Vertex[3]:=55;

Box[No]^.Polygon[42].Vertex[0]:=56;
Box[No]^.Polygon[42].Vertex[1]:=58;
Box[No]^.Polygon[42].Vertex[2]:=59;
Box[No]^.Polygon[42].Vertex[3]:=57;

Box[No]^.Polygon[43].Vertex[0]:=52;
Box[No]^.Polygon[43].Vertex[1]:=58;
Box[No]^.Polygon[43].Vertex[2]:=56;
Box[No]^.Polygon[43].Vertex[3]:=54;

Box[No]^.Polygon[44].Vertex[0]:=53;
Box[No]^.Polygon[44].Vertex[1]:=55;
Box[No]^.Polygon[44].Vertex[2]:=57;
Box[No]^.Polygon[44].Vertex[3]:=59;

Box[No]^.Polygon[45].Vertex[0]:=52;
Box[No]^.Polygon[45].Vertex[1]:=53;
Box[No]^.Polygon[45].Vertex[2]:=59;
Box[No]^.Polygon[45].Vertex[3]:=58;

For A:=20 to 39 do
  For B:=0 to Box[No]^.Polygon[A].TotalVertex-1 do
    Begin
      Inc(Box[No]^.Polygon[A].Vertex[B],26);
    End;

For A:=20 to 39 do
  For B:=0 to (Box[No]^.Polygon[A].TotalVertex-1) div 2 do
    Begin
      Bf:=Box[No]^.Polygon[A].Vertex[B];
      Box[No]^.Polygon[A].Vertex[B]:=Box[No]^.Polygon[A].Vertex[
        Box[No]^.Polygon[A].TotalVertex-1-B];
      Box[No]^.Polygon[A].Vertex[Box[No]^.Polygon;
        [A].TotalVertex-1-B]:=Bf;
    End;

For A:=0 to 45 do Box[No]^.Polygon[A].Color:=120;
For A:=10 to 15 do Box[No]^.Polygon[A].Color:=40;
For A:=16 to 19 do Box[No]^.Polygon[A].Color:=0;
For A:=36 to 39 do Box[No]^.Polygon[A].Color:=0;
For A:=30 to 35 do Box[No]^.Polygon[A].Color:=40;
For A:=41 to 45 do Box[No]^.Polygon[A].Color:=0;
Box[No]^.Polygon[41].Color:=40;
Box[No]^.TotalPoly:=46;
```

```
    Box[No]^.TotalVertex:=60;
    Box[No]^.XR:=X;Box[No]^.YR:=Y;Box[No]^.ZR:=Z;
  End;

Procedure Remove_PolyList;
Var A:Integer;
  Begin
    For A:=0 to 1000 do
      PolygonList[A]^.Active:=False;
    ListCtr:=0;
  End;

Procedure Initialize;
Var A:Integer;
    Height:Longint;
  Begin
    OpenVga256;
    GetMem(Buff,64000);
    Randomize;
    GetMem(Box[0],SizeOf(ObjRect));
    For A:=1 to Total_Obj do
      Begin
        GetMem(Box[A],SizeOf(ObjRect));
        Mat_Identity(Box[A]^.OTMat);
        Mat_Identity(Box[A]^.OIMat);
        Height:=Round(Random(200))+20;
        Create_Box(A,Round(Random(2600))-1300,Height-300,
          Round(Random(8000))-4000,Round(Random(40))+30,Height,
          Round(Random(50))+50,((A mod 5)-1)*40);
      End;
    Mat_Identity(Box[0]^.OTMat);
    Mat_Identity(Box[0]^.OIMat);
    Create_Ship(0,0,0,300);
    Remove_PolyList;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A,63-Round(A*1.5),63-Round(A*1.5),;
          63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+40,63-Round(A*1.5),0,0);
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+80,0,63-Round(A*1.5),0);
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+120,0,0,63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+160,0,63-Round(A*1.5),63-Round(A*1.5));
      End;
    For A:=1 to 40 do
      Begin
        SetVisualPalette(A+200,63-Round(A*1.5),63-Round(A*1.5),0);
      End;
    For A:=0 to 1000 do GetMem(PolygonList[A],SizeOf(PolyList_Rec));
```

```
    With ViewPort do
      Begin
        X:=0;Y:=0;Z:=0;Dx:=0;Dy:=0;Dz:=0;
      End;
    ViewMode:=0;
    Mat_Identity(VMat);
    Mat_Identity(IMat);
  End;

Procedure DeInitialize;
  Begin
    CloseVga256;
  End;

Procedure VertexBank_Projection(No:Integer);
Var A,X1,Y1,X2,Y2:Longint;
  Begin
    With Box[No]^ do
      Begin
        For A:=0 to TotalVertex-1 do
          Begin
            Psv_Projection(VertexBank[A].X+X,VertexBank[A].Y+Y,
              VertexBank[A].Z+Z,VertexBank[A].Xp,VertexBank[A].Yp);
          End;
      End;
  End;

Procedure Assign_Polygon_List(No:Integer);
Var A,B:Integer;
  Begin
    With Box[No]^ do
      Begin
        For A:=0 to TotalPoly-1 do
          Begin
            PolygonList[ListCtr]^.Color:=Polygon[A].Color;
            PolygonList[ListCtr]^.ObjNo:=No;
            PolygonList[ListCtr]^.PolyNo:=A;
            PolygonList[ListCtr]^.Active:=True;
            PolygonList[ListCtr]^.ZMax:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr]^.ZMin:=VertexBank;
               [Polygon[A].Vertex[0]].Z+Z;
            PolygonList[ListCtr]^.XMax:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr]^.XMin:=VertexBank;
               [Polygon[A].Vertex[0]].Xp;
            PolygonList[ListCtr]^.YMax:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            PolygonList[ListCtr]^.YMin:=VertexBank;
               [Polygon[A].Vertex[0]].Yp;
            For B:=0 to Polygon[A].TotalVertex-1 do
              Begin
                If PolygonList[ListCtr]^.ZMax<VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
                   PolygonList[ListCtr]^.ZMax:=VertexBank;
                       [Polygon[A].Vertex[B]].Z+Z;
                If PolygonList[ListCtr]^.ZMin>VertexBank;
                  [Polygon[A].Vertex[B]].Z+Z then
```

```
                    PolygonList[ListCtr]^.ZMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Z+Z;

                  If PolygonList[ListCtr]^.YMax<VertexBank;
                    [Polygon[A].Vertex[B]].Yp then
                     PolygonList[ListCtr]^.YMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Yp;
                  If PolygonList[ListCtr]^.YMin>VertexBank;
                    [Polygon[A].Vertex[B]].Yp then
                     PolygonList[ListCtr]^.YMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Yp;
                  If PolygonList[ListCtr]^.XMax<VertexBank;
                    [Polygon[A].Vertex[B]].Xp then
                     PolygonList[ListCtr]^.XMax:=VertexBank;
                        [Polygon[A].Vertex[B]].Xp;
                  If PolygonList[ListCtr]^.XMin>VertexBank;
                    [Polygon[A].Vertex[B]].Xp then
                     PolygonList[ListCtr]^.XMin:=VertexBank;
                        [Polygon[A].Vertex[B]].Xp;
              End;
            Inc(ListCtr);
          End;
      End;
  End;

Procedure Display_PList_WF;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
    For A:=0 to ListCtr-1 do
     If PolygonList[A]^.Active then
      Begin
        ObjNo :=PolygonList[A]^.ObjNo;
        PolyNo:=PolygonList[A]^.PolyNo;
        With Box[ObjNo]^ do
          Begin
            X1:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y1:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            For B:=1 to Polygon[PolyNo].TotalVertex-1 do
              Begin
                X2:=VertexBank[Polygon[PolyNo].Vertex[B]].Xp;
                Y2:=VertexBank[Polygon[PolyNo].Vertex[B]].Yp;
                Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo]^.Color);
                X1:=X2;         Y1:=Y2;
              End;
            X2:=VertexBank[Polygon[PolyNo].Vertex[0]].Xp;
            Y2:=VertexBank[Polygon[PolyNo].Vertex[0]].Yp;
            Line(Buff^,X1,Y1,X2,Y2,Box[ObjNo]^.Color);
          End;
      End;
  End;

Procedure Poly_Shading(Poly:Poly_Rec;ObjNo,Color:Integer);
Var A           : Integer;
    LeftStart   : Integer;
    RightStart  : Integer;
    RightSlope  : Real;
    LeftSlope   : Real;
    XLeft,XRight: Real;
    RightEnd    : Integer;
    LeftEnd     : Integer;
```

```
  YCtr         : Longint;
  YMin,YMax    : Longint;
  YMinIndex    : Integer;
  YMaxIndex    : Integer;
  RightCtr     : Integer;
  LeftCtr      : Integer;
  XL,XR,Bf     : Longint;
  Dx,Dy        : Real;
  Finish       : Boolean;
  Ty           : Longint;
  Length       : Word;
  Off,Offset   : Word;
Begin
  YMin:=Box[ObjNo]^.VertexBank[Poly.Vertex[0]].Yp;
  YMinIndex:=0;
  YMax:=Box[ObjNo]^.VertexBank[Poly.Vertex[0]].Yp;
  YMaxIndex:=0;

  For A:=1 to Poly.TotalVertex-1 do
    Begin
      Ty:=Box[ObjNo]^.VertexBank[Poly.Vertex[A]].Yp;
      If YMin>Box[ObjNo]^.VertexBank[Poly.Vertex[A]].Yp then
        Begin
          YMin:=Box[ObjNo]^.VertexBank[Poly.Vertex[A]].Yp;
          YMinIndex:=A;
        End;
      If YMax<Box[ObjNo]^.VertexBank[Poly.Vertex[A]].Yp then
        Begin
          YMax:=Box[ObjNo]^.VertexBank[Poly.Vertex[A]].Yp;
          YMaxIndex:=A;
        End;
    End;

  LeftStart:=YMinIndex; RightStart:=YMinIndex;
  YCtr :=YMin;
  LeftEnd:=LeftStart+1;
  If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
  RightEnd:=RightStart-1;
  If RightEnd<0 then RightEnd:=Poly.TotalVertex-1;
  Dx:=Box[ObjNo]^.VertexBank[Poly.Vertex[LeftEnd]].;
     Xp-Box[ObjNo]^.VertexBank[Poly.Vertex[LeftStart]].Xp;
  Dy:=Box[ObjNo]^.VertexBank[Poly.Vertex[LeftEnd]].;
     Yp-Box[ObjNo]^.VertexBank[Poly.Vertex[LeftStart]].Yp+1;
  Finish:=False;
  While (Dy=0) and (Not Finish) do
    Begin
      LeftStart:=LeftStart+1;
      If LeftStart>=Poly.TotalVertex then LeftStart:=0;
      LeftEnd:=LeftStart+1;
      Dx:=Box[ObjNo]^.VertexBank[Poly.Vertex[LeftEnd]].;
        Xp-Box[ObjNo]^.VertexBank[Poly.Vertex[LeftStart]].Xp;
      Dy:=Box[ObjNo]^.VertexBank[Poly.Vertex[LeftEnd]].;
        Yp-Box[ObjNo]^.VertexBank[Poly.Vertex[LeftStart]].Yp+1;
      If LeftStart=YMaxIndex then Finish:=True;
    End;
  If Not Finish then
    Begin
      LeftSlope:=(Dx)/(Dy);
      Dx:=Box[ObjNo]^.VertexBank[Poly.Vertex[RightEnd]].;
        Xp-Box[ObjNo]^.VertexBank[Poly.Vertex[RightStart]].Xp;
```

```
Dy:=Box[ObjNo]^.VertexBank[Poly.Vertex[RightEnd]].;
   Yp-Box[ObjNo]^.VertexBank[Poly.Vertex[RightStart]].Yp+1;
While (Dy=0) and (Not Finish) do
  Begin
    RightStart:=RightStart-1;
    If RightStart<0 then RightStart:=Poly.TotalVertex-1;
    RightEnd:=RightStart-1;
    Dx:=Box[ObjNo]^.VertexBank[Poly.Vertex[RightEnd]].;
       Xp-Box[ObjNo]^.VertexBank;
        [Poly.Vertex[RightStart]].Xp;
    Dy:=Box[ObjNo]^.VertexBank[Poly.Vertex[RightEnd]].;
       Yp-Box[ObjNo]^.VertexBank;
        [Poly.Vertex[RightStart]].Yp+1;
    If RightStart=YMaxIndex then Finish:=True;
  End;
RightSlope:=(Dx)/(Dy);
XLeft :=Box[ObjNo]^.VertexBank;
     [Poly.Vertex[LeftStart]].Xp+LeftSlope;
XRight:=Box[ObjNo]^.VertexBank;
     [Poly.Vertex[RightStart]].Xp+RightSlope;
Off:=YCtr*320;
While (Not Finish) do
  Begin
    XL:=Round(XLeft);
    XR:=Round(XRight);

    If XL>XR then Finish:=True;

    If XL<0 then XL:=0;
    If XR>319 then XR:=319;
    If (YCtr>=0) and (Not Finish) then
     If XL=XR then {PutPixel(Buff^,XL,YCtr,Color)} else
      If XL<XR then
        Begin
          Length:=XR-XL+1;
          Offset:=Off+XL;
          FillLines(Buff^,Offset,Length,Color);
        End;
    XLeft:=XLeft+LeftSlope;
    XRight:=XRight+RightSlope;
    Inc(YCtr);
    Inc(Off,320);
    If YCtr>199 then Finish:=True;
    If Not Finish then
     If (YCtr>Box[ObjNo]^.VertexBank;
       [Poly.Vertex[LeftEnd]].Yp) then
      Begin
        Repeat
          LeftStart:=LeftStart+1;
          If LeftStart>=Poly.TotalVertex then LeftStart:=0;
          LeftEnd:=LeftStart+1;
          If LeftEnd>=Poly.TotalVertex then LeftEnd:=0;
          Dx:=Box[ObjNo]^.VertexBank;
            [Poly.Vertex[LeftEnd]].;
              Xp-Box[ObjNo]^.VertexBank;
               [Poly.Vertex[LeftStart]].Xp;
          Dy:=Box[ObjNo]^.VertexBank;
            [Poly.Vertex[LeftEnd]].;
              Yp-Box[ObjNo]^.VertexBank;
               [Poly.Vertex[LeftStart]].Yp+1;
```

```
                If LeftStart=YMaxIndex then Finish:=True;
              Until (Dy<>0) or Finish;
              If Not Finish then
                Begin
                  LeftSlope:=(Dx)/(Dy);
                  XLeft:=Box[ObjNo]^.VertexBank;
                     [Poly.Vertex[LeftStart]].Xp+LeftSlope;
                End;
            End;
          If Not Finish then
            Begin
              If YCtr>Box[ObjNo]^.VertexBank;
                [Poly.Vertex[RightEnd]].Yp then
                Begin
                  Repeat
                    RightStart:=RightStart-1;
                    If RightStart<0 then;
                     RightStart:=Poly.TotalVertex-1;
                    RightEnd:=RightStart-1;
                    If RightEnd<0 then;
                     RightEnd:=Poly.TotalVertex-1;
                    Dx:=Box[ObjNo]^.VertexBank;
                     [Poly.Vertex[RightEnd]].;
                        Xp-Box[ObjNo]^.VertexBank;
                            [Poly.Vertex[RightStart]].Xp;
                    Dy:=Box[ObjNo]^.VertexBank;
                     [Poly.Vertex[RightEnd]].;
                        Yp-Box[ObjNo]^.VertexBank;
                            [Poly.Vertex[RightStart]].Yp
                    +1;
                    If RightStart=YMaxIndex then Finish:=True;
                  Until (Dy<>0) or Finish;
                  If Not Finish then
                    Begin
                      RightSlope:=(Dx)/(Dy);
                      XRight:=Box[ObjNo]^.VertexBank;
                          [Poly.Vertex[RightStart]].Xp;
                      XRight:=XRight+RightSlope;
                    End;
                End;
            End;
        End;
    End;
  End;

Function Vertical_VV_Testing(YMin,YMax:Real) : Boolean;
  Begin
    Vertical_VV_Testing:=False;
    If (YMin>199) or (YMax<0) then Vertical_VV_Testing:=True;
  End;

Function Horisontal_VV_Testing(XMin,XMax:Real) : Boolean;
  Begin
    Horisontal_VV_Testing:=False;
    If (XMin>319) or (XMax<0) then Horisontal_VV_Testing:=True;
  End;

Procedure Display_PList_Shading;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
```

```
    ColCtr:=17;
    For A:=0 to ListCtr-1 do
     If PolygonList[A]^.Active then
      Begin
        ObjNo :=PolygonList[A]^.ObjNo;
        PolyNo:=PolygonList[A]^.PolyNo;
        Poly_Shading(Box[ObjNo]^.Polygon;
          [PolyNo],ObjNo,PolygonList[A]^.Color);
      End;
  End;

Procedure View_Volume_Filter;
Var A,B:Integer;
  Begin
    For A:=0 to ListCtr-1 do
      If PolygonList[A]^.Active then
        Begin
          If PolygonList[A]^.ZMax<=100 then;
             PolygonList[A]^.Active:=False;
          If PolygonList[A]^.ZMin<=50  then;
             PolygonList[A]^.Active:=False;
          If Vertical_VV_Testing;
            (PolygonList[A]^.YMin,PolygonList[A]^.YMax)
            then PolygonList[A]^.Active:=False;
          If Horisontal_VV_Testing;
            (PolygonList[A]^.XMin,PolygonList[A]^.XMax)
            then PolygonList[A]^.Active:=False;
        End;
  End;

Procedure BackFace_Filter;
Var View,A,B,Normal :Vec3D;
    Ctr,ObjNo,PolyNo:Integer;
    Deg              :Real;
    Intensity        :Integer;
  Begin
    View[0]:=0;View[1]:=0;View[2]:=10;
    For Ctr:=0 to ListCtr-1 do
      Begin
        ObjNo :=PolygonList[Ctr]^.ObjNo;
        PolyNo:=PolygonList[Ctr]^.PolyNo;
        A[0]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[0]].X-
               Box[ObjNo]^.VertexBank;
                 [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].X;
        A[1]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[0]].Y-
               Box[ObjNo]^.VertexBank;
                 [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].Y;
        A[2]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[0]].Z-
               Box[ObjNo]^.VertexBank;
                 [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].Z;

        View[0]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].X;
        View[1]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].Y;
        View[2]:=-Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].Z-View_Distance;
```

```
        B[0]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[2]].X-
               Box[ObjNo]^.VertexBank;
                 [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].X;
        B[1]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[2]].Y-
               Box[ObjNo]^.VertexBank;
                 [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].Y;
        B[2]:=Box[ObjNo]^.VertexBank;
         [Box[ObjNo]^.Polygon[PolyNo].Vertex[2]].Z-
               Box[ObjNo]^.VertexBank;
                 [Box[ObjNo]^.Polygon[PolyNo].Vertex[1]].Z;
        Vect_Cross(Normal,A,B);
        Deg:=Abs(Abs(Vect_Angle(Normal,View)));
        Intensity:=Round(Deg/2.5);
        PolygonList[Ctr]^.Color:=Box[ObjNo]^.Polygon;
         [PolygonList[Ctr]^.PolyNo].Color+Intensity+1;
      End;
  End;

Procedure Display_Try;
Var A,ObjNo,PolyNo,B,X1,Y1,X2,Y2,ColCtr:Integer;
  Begin
        ObjNo :=0;
        PolyNo:=0;
        Box[ObjNo]^.VertexBank[0].Xp:=160;
        Box[ObjNo]^.VertexBank[0].Yp:=30;
        Box[ObjNo]^.VertexBank[1].Xp:=220;
        Box[ObjNo]^.VertexBank[1].Yp:=100;
        Box[ObjNo]^.VertexBank[2].Xp:=160;
        Box[ObjNo]^.VertexBank[2].Yp:=130;
        Box[ObjNo]^.VertexBank[3].Xp:=60;
        Box[ObjNo]^.VertexBank[3].Yp:=100;
        Box[ObjNo]^.Polygon[0].Vertex[0]:=0;
        Box[ObjNo]^.Polygon[0].Vertex[1]:=3;
        Box[ObjNo]^.Polygon[0].Vertex[2]:=2;
        Box[ObjNo]^.Polygon[0].Vertex[3]:=1;
        Box[ObjNo]^.Polygon[0].TotalVertex:=4;
        Box[ObjNo]^.TotalPoly:=1;
        Box[ObjNo]^.TotalVertex:=4;
        Poly_Shading(Box[0]^.Polygon[0],0,15);
  End;

Function Appr_View_Volume_Filter(No:Integer):Boolean;
Var X,Y:Longint;
  Begin
    Appr_View_Volume_Filter:=True;
    If (Box[No]^.Z<100) or (Box[No]^.Z>5000) then;
       Appr_View_Volume_Filter:=False else
      Begin
        Psv_Projection(Box[No]^.X,Box[No]^.Y,Box[No]^.Z,X,Y);
        If (X<-300) or (X>600) or (Y<-200) or (Y>400) then;
          Appr_View_Volume_Filter:=False;
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to Total_Obj do
```

```
      Begin
        If Appr_View_Volume_Filter(A) then
          Begin
            VertexBank_Projection(A);
            Assign_Polygon_List(A);
          End;
      End;
  End;

Procedure ViewPort_Transform;
Var A,B:Integer;
    CMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    For A:=0 to Total_Obj do
      With Box[A]^ do
        Begin
          For B:=0 to TotalVertex-1 do
            With VertexBank[B] do
              Begin
                CMat[0]:=XR;
                CMat[1]:=YR;
                CMat[2]:=ZR;
                CMat[3]:=1;
                Mat_Multiply4x1(OTMat,CMat);
                Mat_Multiply4x1(IMat,CMat);
                X:=CMat[0];
                Y:=CMat[1];
                Z:=CMat[2];
              End;
          CMat[0]:=XR-ViewPort.X;
          CMat[1]:=YR-ViewPort.Y;
          CMat[2]:=ZR-ViewPort.Z;
          CMat[3]:=1;
          Mat_Multiply4x1(IMat,CMat);
          X:=CMat[0];
          Y:=CMat[1];
          Z:=CMat[2];
        End;
  End;

Procedure Transform(No:Integer;A:Mat4x4);
Var d:Integer;
    CMat:Mat1x4;
  Begin
    With Box[No]^ do
      Begin
        For d:=0 to TotalVertex-1 do
          Begin
            CMat[0]:=VertexBank[d].XR;
            CMat[1]:=VertexBank[d].YR;
            CMat[2]:=VertexBank[d].ZR;
            CMat[3]:=1;
            Mat_Multiply4x1(A,CMat);
            VertexBank[d].XR:=CMat[0];
            VertexBank[d].YR:=CMat[1];
            VertexBank[d].ZR:=CMat[2];
          End;
      End;
  End;
```

```
Procedure X_Rotation(Deg:Longint);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_XRotate(XMat,Deg);
    Mat_Multiply4x4(Box[0]^.OTMat,XMat);
    Mat_XRotate(XMat,-Deg);
    Mat_Multiply4x4(XMat,Box[0]^.OIMat);
    Box[0]^.OIMat:=XMat;
  End;

Procedure Y_Rotation(Deg:Longint);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_YRotate(YMat,Deg);
    Mat_Multiply4x4(Box[0]^.OTMat,YMat);
    Mat_YRotate(YMat,-Deg);
    Mat_Multiply4x4(YMat,Box[0]^.OIMat);
    Box[0]^.OIMat:=YMat;
  End;

Procedure Y_Rotation2(Deg:Longint);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_YRotate(YMat,Deg);
    Mat_Multiply4x4(YMat,Box[0]^.OTMat);
    Box[0]^.OTMat:=YMat;
    Mat_YRotate(YMat,-Deg);
    Mat_Multiply4x4(Box[0]^.OIMat,YMat);
  End;

Procedure Z_Rotation(Deg:Longint);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_ZRotate(ZMat,Deg);
    Mat_Multiply4x4(Box[0]^.OTMat,ZMat);
    Mat_ZRotate(ZMat,-Deg);
    Mat_Multiply4x4(ZMat,Box[0]^.OIMat);
    Box[0]^.OIMat:=ZMat;
  End;

Procedure Scale(Sx,Sy,Sz:Real);
Var SMat:Mat4x4;
    A:Integer;
  Begin
    Mat_Scaling(SMat,Sx,Sy,Sz);
    For A:=0 to Total_Obj do
      Transform(A,SMat);
  End;

Procedure Dilatasi(Tx,Ty,Tz:Real);
Var A:Integer;
```

```
  Begin
    For A:=0 to Total_Obj do
      Begin
        Box[A]^.XR:=Box[A]^.XR+Tx;
        Box[A]^.YR:=Box[A]^.YR+Ty;
        Box[A]^.ZR:=Box[A]^.ZR+Tz;
      End;
  End;

Procedure Painter_Sort;
Var A,B:Integer;
    Bf :PolyList_Rec;
  Begin
    For A:=0 to ListCtr-2 do
     If PolygonList[A]^.active then
      For B:=A+1 to ListCtr-1 do
        If PolygonList[B]^.active then
          If PolygonList[A]^.ZMin<PolygonList[B]^.ZMin then
            Begin
              Bf:=PolygonList[A]^;
              PolygonList[A]^:=PolygonList[B]^;
              PolygonList[B]^:=Bf;
            End;
  End;

Procedure Draw_Map;
Var X,Y:Longint;
    A:Integer;
  Begin
    Rectangle(Buff^,10,10,110,100,15);
    Block(Buff^,11,11,109,99,0);
    Line(Buff^,60,50,60,60,14);
    Line(Buff^,55,55,65,55,14);
    X:=Round(ViewPort.X/20)+60;
    Y:=-Round(ViewPort.Z/20)+55;
    Rectangle(Screen^,X-2,Y-2,X+2,Y+2,30);
    For A:=0 to Total_Obj do
      Begin
        PutPixel(Buff^,Round(Box[A]^.XR/20)+60,;
        -Round(Box[A]^.ZR/20)+55,15);
      End;
    Rectangle(Buff^,10,105,110,195,15);
    Block(Buff^,11,106,109,194,0);
    Line(Buff^,60,145,60,155,14);
    Line(Buff^,55,150,65,150,14);
    X:=Round(ViewPort.X/20)+60;
    Y:=-Round(ViewPort.Y/20)+150;
    Rectangle(Screen^,X-2,Y-2,X+2,Y+2,30);
    For A:=0 to Total_Obj do
      Begin
        PutPixel(Buff^,Round(Box[A]^.XR/20)+60,;
        -Round(Box[A]^.YR/20)+150,15);
      End;
  End;

Procedure Move_Object(Dir:Integer);
Var BMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    BMat[0]:=0;
```

```
    BMat[1]:=0;
    BMat[2]:=4*Dir;
    BMat[3]:=1;
    Mat_Multiply4x1(Box[0]^.OTMat,BMat);
    Box[0]^.XR:=Box[0]^.XR+BMat[0];
    Box[0]^.YR:=Box[0]^.YR+BMat[1];
    Box[0]^.ZR:=Box[0]^.ZR+BMat[2];
  End;

Procedure Rotate_X(Dir:Integer);
Var XMat:Mat4x4;
    CMat:Mat1x4;
  Begin
    Mat_XRotate(XMat,Dir);
    Mat_Multiply4x4(VMat,XMat);
    Mat_XRotate(XMat,-Dir);
    Mat_Multiply4x4(XMat,IMat);
    IMat:=XMat;
  End;

Procedure Rotate_Y(Dir:Integer);
Var XMat:Mat4x4;
    CMat:Mat1x4;
  Begin
    Mat_YRotate(XMat,Dir);
    Mat_Multiply4x4(VMat,XMat);
    Mat_YRotate(XMat,-Dir);
    Mat_Multiply4x4(XMat,IMat);
    IMat:=XMat;
  End;

Procedure Rotate_Z(Dir:Integer);
Var XMat:Mat4x4;
    CMat:Mat1x4;
  Begin
    Mat_ZRotate(XMat,Dir);
    Mat_Multiply4x4(VMat,XMat);
    Mat_ZRotate(XMat,-Dir);
    Mat_Multiply4x4(XMat,IMat);
    IMat:=XMat;
  End;

Procedure Adjust_ViewPort(No:Integer);
Var BMat:Mat1x4;
    XMat,YMat,ZMat:Mat4x4;
  Begin
    If ViewMode<>0 then
      Begin
        If CtrView<FixViewDistance then
          Begin
            CtrView:=CtrView+20;
            If CtrView>FixViewdistance then;
               CtrView:=FixViewdistance;
          End;
        If CtrView>FixViewDistance then
          Begin
            CtrView:=CtrView-20;
            If CtrView<FixViewdistance then;
               CtrView:=FixViewdistance;
          End;
```

```
        BMat[0]:=0;
        BMat[1]:=20;
        BMat[2]:=-CtrView;
        BMat[3]:=1;
        Mat_Multiply4x1(Box[0]^.OTMat,BMat);
        ViewPort.X:=Box[0]^.XR+BMat[0];
        ViewPort.Y:=Box[0]^.YR+BMat[1];
        ViewPort.Z:=Box[0]^.ZR+BMat[2];
        VMat:=Box[0]^.OTMat;
        IMat:=Box[0]^.OIMat;
      End;
  End;

Procedure Main;
Var Ch:Char;
    Speed:Integer;
    DZ:Integer;
    DY:Integer;
    DX:Integer;
  Begin
    Ch:='s';
    Speed:=0;
    DZ:=0;
    DY:=0;
    DX:=0;
    CtrView:=50;
    Repeat
      Repeat
        FillWord(Buff^,32000,0);
        Remove_PolyList;
        ViewPort_Transform;
        Project_Object;
        View_Volume_Filter;
        BackFace_Filter;
        Painter_Sort;
        Display_PList_Shading;
        MoveTo(Buff^,Screen^);
        Case ch of
          #80:Begin
                If DX<30 then
                  Begin
                    X_Rotation(4);
                    Inc(DX,4);
                  End;
              End;
          #72:Begin
                If DX>-30 then
                  Begin
                    X_Rotation(-4);
                    Dec(DX,4);
                  End;
              End;
          #75:Begin
                If DZ>-30 then
                  Begin
                    Z_Rotation(-4);
                    Dec(DZ,4);
                  End;
              End;
          #77:Begin
                If DZ<30 then
```

```
          Begin
            Z_Rotation(4);
            Inc(DZ,4);
          End;
      End;
  'r':X_Rotation(2);
  'f':X_Rotation(-2);
  'a':Speed:=2;
  'z':Speed:=-2;
  ' ':Begin
        If DX<0 then
          Begin
            If DX>-4 then
              Begin
                X_Rotation(-DX);
                DX:=0;
              End Else
              Begin
                X_Rotation(4);
                Inc(DX,4);
              End;
          End;
        If DX>0 then
          Begin
            If DX<4 then
              Begin
                X_Rotation(-DX);
                DX:=0;
              End Else
              Begin
                X_Rotation(-4);
                Dec(DX,4);
              End;
          End;
        If DZ<0 then
          Begin
            If DZ>-3 then
              Begin
                Z_Rotation(-DZ);
                DZ:=0;
              End Else
              Begin
                Z_Rotation(3);
                Inc(DZ,3);
              End;
          End;
        If DZ>0 then
          Begin
            If DZ<3 then
              Begin
                Z_Rotation(-DZ);
                DZ:=0;
              End Else
              Begin
                Z_Rotation(-3);
                Dec(DZ,3);
              End;
          End;
      End;
End;
If DZ<0 then
```

```
          Begin
            Y_Rotation2(2);
          End;
        If DZ>0 then
          Begin
            Y_Rotation2(-2);
          End;
        Move_Object(Speed);
        Adjust_ViewPort(ViewMode);
      Until Keypressed;
      Ch:=Readkey;
      if Ch=#13 then Inc(ViewMode);
      FixViewdistance:=ViewMode*100-50;
      If ViewMode>4 then ViewMode:=0;
    Until Ch=#27;
  End;

Begin
  Initialize;
  Main;
  DeInitialize;
End.
```

PENJELASAN PROGRAM

Coba jalankan program di atas. WOW!!! Kecepatan animasi bertambah lagi. Perhatikan jumlah objek pada dunia tiga dimensi. Pada program ini digunakan objek sejumlah 50 buah. Bandingkan kecepatannya dengan program B3FLGHT.PAS yang cuma menampilkan 10 objek saja.

PENJELASAN LISTING PROGRAM

Marilah kita lihat bagian yang telah dimodifikasi.

```
Var A,B:Integer;
```

Box dan PolygonList diubah menjadi pointer untuk mengurangi jumlah pemakaian program segment memori yang berjumlah 64 kbytes. Hal ini disebabkan karena kita menggunakan polygon list dan objek lebih banyak.

```
Box:array[0..Total_Obj] of ^ObjRect;
PolygonList:array[0..1000] of ^PolyList_Rec;
ViewPort:ViewPort_Rec;
ListCtr:Integer;
buff:Pointer;
VMat:Mat4x4;
IMat:Mat4x4;
ViewMode:Integer;
CtrView,FixViewDistance:Longint;
```

Perhatikan pada procedure X_Rotation, Y_Rotation, Y_Rotation2 dan Z_Rotation dilakukan perkalian Deg dengan 100 karena sudut 1 derajat di MATH3.PAS berarti 100 derajat.

```
Procedure X_Rotation(Deg:Longint);
Var XMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_XRotate(XMat,Deg);
    Mat_Multiply4x4(Box[0]^.OTMat,XMat);
    Mat_XRotate(XMat,-Deg);
    Mat_Multiply4x4(XMat,Box[0]^.OIMat);
    Box[0]^.OIMat:=XMat;
  End;

Procedure Y_Rotation(Deg:Longint);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_YRotate(YMat,Deg);
    Mat_Multiply4x4(Box[0]^.OTMat,YMat);
    Mat_YRotate(YMat,-Deg);
    Mat_Multiply4x4(YMat,Box[0]^.OIMat);
    Box[0]^.OIMat:=YMat;
  End;

Procedure Y_Rotation2(Deg:Longint);
Var YMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_YRotate(YMat,Deg);
    Mat_Multiply4x4(YMat,Box[0]^.OTMat);
    Box[0]^.OTMat:=YMat;
    Mat_YRotate(YMat,-Deg);
    Mat_Multiply4x4(Box[0]^.OIMat,YMat);
  End;

Procedure Z_Rotation(Deg:Longint);
Var ZMat:Mat4x4;
    A:Integer;
  Begin
    Deg:=Deg*100;
    Mat_ZRotate(ZMat,Deg);
    Mat_Multiply4x4(Box[0]^.OTMat,ZMat);
    Mat_ZRotate(ZMat,-Deg);
    Mat_Multiply4x4(ZMat,Box[0]^.OIMat);
    Box[0]^.OIMat:=ZMat;
  End;
```

Perhatikan cuplikan dari fungsi POLY_SHADING.

Di sini dilakukan pemanggilan fungsi FillLines sebagai pengganti HLines. Karena dengan memanggil HLines setiap kita melakukan pengarsiran, selalu dilakukan perhitungan offset yang melibatkan sejumlah SHL. Untuk itu dilakukan proses perhitungan offset secara langsung dan dikirimkan lewat parameter pada FillLines. Length ialah panjang garis yang hendak digambar.

```
If XL<XR then
                Begin
                  Length:=XR-XL+1;
                  Offset:=Off+XL;
                  FillLines(Buff^,Offset,Length,Color);
                End;
```

Procedure di bawah ini dipanggil untuk mengurangi jumlah poligon pada polygon list.

```
Function Appr_View_Volume_Filter(No:Integer):Boolean;
Var X,Y:Longint;
  Begin
```

Asumsikan objek berada di dalam view volume.

```
    Appr_View_Volume_Filter:=True;
```

Dilakukan pendeteksian koordinat semu Z dari objek.

Jika di luar view volume maka kembalikan harga FALSE.

```
    If (Box[No]^.Z<100) or (Box[No]^.Z>5000) then;
       Appr_View_Volume_Filter:=False else
      Begin
```

Jika objek berada di view volume berdasarkan testing pertama, lakukan testing kedua yaitu deteksi koordinat X dan Y hasil transformasi. Di sini dilakukan perkiraan batas layar yaitu (-300,-200) hingga (600,400). Karena ada kemungkinan objek yang berada di luar layar monitor masih terlihat sepotong, sehingga diperlukan pendeteksian terhadap area yang lebih besar sedikit dari layar monitor.

```
        Psv_Projection(Box[No]^.X,Box[No]^.Y,Box[No]^.Z,X,Y);
        If (X<-300) or (X>600) or (Y<-200) or (Y>400) then;
          Appr_View_Volume_Filter:=False;
      End;
  End;

Procedure Project_Object;
Var A:Integer;
  Begin
    For A:=0 to Total_Obj do
      Begin
```

Jika objek diperkirakan berada dalam view volume maka lakukan proyeksi dan pendaftaran poligon-poligonnya dalam polygon list.

```
        If Appr_View_Volume_Filter(A) then
          Begin
            VertexBank_Projection(A);
            Assign_Polygon_List(A);
          End;
      End;
  End;
```

Sebenarnya masih ada beberapa teknik optimasi lagi yang dapat diterapkan pada program di atas. Seperti pendeteksian element 0 terhadap fungsi Mat_Multiply4x1 seperti yang telah kita lakukan pada Mat_Multiply4x4. Juga pendeteksian terhadap angka 1 pada element matrix karena angka 1 juga merupakan angka favorit yang sering dijumpai pada matrix transformasi dan hasil perkalian dengan 1 ialah bilangan itu sendiri. Maka kita dapat melakukan pengurangan perkalian lagi.

Hal lain ialah dengan mengganti seluruh sistem bilangan Real dengan bilangan Longint dan menerapkan teknik FIXED POINT ALGORITHM. (Pada buku Seri Aplikasi Pemrograman 'Pemrograman Animasi dan Game Profesional 2'.)

Tetapi penggunaan teknik ini bisa dibilang sangat rumit apalagi dalam pemrograman tiga dimensi. Jika Anda benar-benar ingin membuat sebuah game tiga dimensi yang dapat bersaing di pasaran game internasional sekarang ini dengan standard processor 486 ke bawah, Anda harus menggunakan teknik ini. Kita tidak perlu khawatir akan pemakaian FLOATING POINT jika standar processor yang dipakai ialah PENTIUM karena menurut desas-desus, FLOATING POINT dari Pentium lebih cepat dari operasi FIXED POINT-nya. Hal ini telah dibuktikan oleh game QUAKE yang dibuat dengan Floating Point dan menggunakan standar PENTIUM.

Ubah proses sorting yang dilakukan menjadi sorting object, bukannya sorting per-poligon karena objek yang kita gunakan masih sederhana yaitu balok. Atau gunakan fungsi sorting yang lebih baik seperti QSORT, HEAP SORT yang memiliki performance yang bagus untuk jumlah data yang cukup banyak seperti pada program kita kali ini.

Mengingat objek-objek selain pesawat kita pada dunia tiga dimensi tidak bergerak, kita dapat menerapkan Binary Space Data Structure (dengan dasar Binary Tree) dalam penyimpanan poligon dari sebuah objek sehingga tidak perlu dilakukannya proses sorting.

Satu lagi optimasi yang dapat Anda lakukan, konversikan seluruh program ke ASSEMBLER (he-he-he). Yang terakhir ini benar-benar siksaan yang

sangat berat. (Kalau mau dicoba sih boleh-boleh saja kok.)

Optimasi-optimasi lebih lanjut saya serahkan kepada para pembaca.

Ok, sampai di sinilah pembahasan kita kali ini. Pada kesempatan yang akan datang akan kita bahas pembuatan game tiga dimensi dengan menggunakan RAY TRACING seperti yang dilakukan DOOM, Wolfenstein 3D juga teknik VOXEL SPACE yang digunakan untuk merakit Comanche, Magic Carpet dan teknik-teknik lainnya, yang pokoknya SIIIIP deh!!! Bye-bye!!!

BAB 10

ULASAN TENTANG VGACAD.PAS

Pada bab ini akan diberikan servis-servis yang disediakan pada VGACAD.PAS yang telah dimodifikasi lebih lanjut setelah mengalami dua kali modifikasi pada buku Seri Aplikasi Pemrograman 'Pemrograman Animasi dan Game Profesional 1 dan 2'.

10.1 Daftar Fungsi dan Procedure dari VGACAD.PAS

Procedure Animasi Tiga Layar

Procedure : FillWord (Var PicBuff; Size:Word; Code:Byte);
Fungsi : Mengisi virtual screen dengan sebuah warna secara WORD.
Parameter : PicBuff = Alamat virtual screen yang dituju.
Size = Jumlah word yang akan diisikan.
Code = Warna yang akan Anda pakai.

Procedure : MoveW (Var Source; Var Dest; Size:Word);
Fungsi : Memindahkan virtual screen ke virtual screen yang lain.
Parameter : Source = Alamat virtual screen asal.
Dest = Alamat virtual screen tujuan.
Size = Besar dari virtual screen yang dipakai.

Procedure : Define_Page (Var PicBuff : Pointer);
Fungsi : Menciptakan sebuah virtual screen dengan alamat PicBuff.
Parameter : PicBuff = Output yang menunjukkan alamat memori yang digunakan oleh virtual screen tersebut.

Procedure : UnDefine_Page (Var PicBuff : Pointer);
Fungsi : Melenyapkan sebuah virtual screen dengan alamat PicBuff.
Parameter : PicBuff = Alamat virtual screen yang akan dilenyapkan.

Procedure : Fill_Page (Var PicBuff; Col : Byte);
Fungsi : Untuk menghapus sebuah virtual screen dengan satu warna.
Parameter : PicBuff = Alamat virtual screen yang dituju.
Col = Warna yang akan diisikan.

Procedure : MoveTo (Var Source, Destination);
Fungsi : Memindahkan sebuah virtual screen ke virtual screen yang lain.
Parameter : Source = Alamat virtual screen asal.
Destination = Alamat virtual screen tujuan.

Procedure Konstruksi Bangun Geometri

Procedure : PutPixel (Var PicBuff; X,Y : Integer; Color : Byte);
Fungsi : Menciptakan titik pada virtual screen.
Parameter : PicBuff = Alamat virtual screen yang dituju.
X,Y = Koordinat dari titik.
Col = Warna dari titik.

Function : GetPixel (Var PicBuff; X,Y : Integer) : Byte;
Fungsi : Membaca sebuah titik pada virtual screen.
Parameter : PicBuff = Alamat virtual screen yang dituju.
X,Y = Koordinat dari titik.
Return : Warna dari titik.

Procedure : Line (Var PicBuff; X1,Y1,X2,Y2:integer; Col : Byte);
Fungsi : Menciptakan garis pada virtual screen.
Parameter : PicBuff = Alamat virtual screen yang dituju.
X1,Y1 = Koordinat start awal dari garis.
X2,Y2 = Koordinat akhir dari garis.
Col = Warna dari garis.

Procedure : VLine (Var PicBuff; X,Y,L:integer; Col : Byte);
Fungsi : Menciptakan garis vertikal pada virtual screen.
Parameter : X,Y = Koordinat awal dari garis.

L = Panjang vertikal dari garis.
Col = Warna dari garis.

Procedure : Dot_Line (Var PicBuff; X1,Y1,X2,Y2:integer; Col : Byte;No:Integer;Tot_Dot:Integer);
Fungsi : Menciptakan garis putus-putus pada virtual screen.
Parameter : PicBuff = Alamat dari virtual screen.
X1,Y1,X2,Y2 = Start awal dan akhir dari garis.
Col = Warna dari garis.
No = Nomor titik yang akan dilingkari dengan kotak.
Tot_Dot = Jumlah titik yang akan ditampilkan sepanjang garis tersebut.

Procedure : Rectangle (Var PicBuffer; X1,Y1,X2,Y2 : Integer; Color : Byte);
Fungsi : Menggambar persegi panjang pada virtual screen.
Parameter : Picbuffer = Alamat dari virtual screen.
X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah dari persegi panjang.
Col = Warna dari persegi panjang.

Procedure : Block (Var PicBuff; X1,Y1,X2,Y2 : Integer; Col : Byte);
Fungsi : Menampilkan balok berisi pada virtual screen.
Parameter : PicBuff = Alamat dari virtual screen.
X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah dari balok.
Col = Warna dari balok.

Procedure : Circle (Var PicBuff; X,Y : Integer; R : Integer; Col : Byte);
Fungsi : Menggambar lingkaran pada virtual screen.
Parameter : PicBuff = Alamat dari virtual screen.
X,Y = Koordinat pusat dari lingkaran.
R = Jari-jari dari lingkaran.
Col = Warna dari lingkaran.

Procedure : Ellipse (Var PicBuff; X,Y : Integer; Rx,Ry : Integer; Col : Byte);
Fungsi : Menggambar elips pada virtual screen.
Parameter : PicBuff = Alamat dari virtual screen.
X,Y = Koordinat pusat dari elips.
Rx, Ry = Jari-jari sumbu X dan Y.
Col = Warna dari elips.

Procedure : VLines (Var PicBuffer; Y1,Y2.X : Integer; Color : Byte);
Fungsi : Menggambar garis vertikal pada virtual screen
Parameter : PicBuffer = Alamat dari virtual screen.
Y1,Y2 = Koordinat vertikal awal dan akhir dari garis.
X = Koordinat horizontal dari garis.
Color = Warna dari elips.

Procedure : HLines (Var PicBuffer; X1,X2.Y : Integer; Color : Byte);
Fungsi : Menggambar garis horisontal pada virtual screen.
Parameter : PicBuffer = Alamat dari virtual screen.
X1,X2 = Koordinat horizontal awal dan akhir dari garis.
Y = Koordinat vertikal dari garis.
Color = Warna dari elips.

Procedure : FillLines (Var PicBuffer; XL,Length:Word; Color : Byte);
Fungsi : Menggambar garis vertikal pada virtual screen
Parameter : PicBuffer = Alamat dari virtual screen.
XL = Offset start peletakkan dari garis.
Length = Panjang dari garis.
Color = Warna dari elips.

Procedure Manipulasi Sprite

Function : ImageSize (X1,Y1,X2,Y2 : Integer) : Word;
Fungsi : Menghitung jumlah memori yang dibutuhkan oleh sprite.
Parameter : X1,Y1,X2,Y2 = Ukuran dari sprite.
Return : Jumlah memori yang diperlukan.

Procedure : GetImage (Var PicBuff; X1,Y1,X2,Y2 : Integer; Var Buffer);
Fungsi : Mengambil sprite dari virtual screen ke memori.
Parameter : PicBuff = Alamat virtual screen.
X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah dari sprite yang akan diambil.
Buffer = Alamat memori tempat sprite disimpan.

Procedure : Load_Sprite (File_Name:String; Var Ptr:Pointer;Var Size:Word);
Fungsi : Membaca sprite dari file.
Parameter : File_Name = Nama file tempat sprite disimpan.
Ptr = Alamat memori tempat sprite akan disimpan.
Size = Output yang menunjukkan besar dari sprite yang dibaca dari file.

Procedure : Save_Sprite (File_Name:String; Ptr:Pointer;Size:Word);
Fungsi : Menyimpan sprite ke dalam file.
Parameter : File_Name = Nama file tempat sprite akan disimpan.
Ptr = Alamat memori tempat sprite disimpan.
Size = Besar dari sprite yang akan disimpan dalam file.

Procedure : Load_Picture (File_Name:String; Var Buffer);
Fungsi : Untuk menampilkan sebuah gambar dari file pada virtual screen.
Parameter : File_Name = Nama file tempat gambar disimpan.
Buffer = Alamat dari virtual screen.

Procedure : Save_Picture (File_Name:String; Var Buffer);
Fungsi : Menyimpan gambar pada virtual screen ke file.
Parameter : File_Name = Nama file tempat gambar akan disimpan.
Buffer = Alamat virtual screen di mana gambar berada.

Procedure : PutOld (Var PicBuff;X,Y : Integer; Var Buffer);
Fungsi : Menampilkan sprite dengan tidak mempedulikan batasan layar.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat di mana kita akan menampilkan sprite.
Buffer = Alamat memori tempat sprite disimpan.

Procedure : PutImage (Var PicBuff;X1,Y1:Integer;Var Buffer);
Fungsi : Menampilkan sprite dengan mempedulikan batasan layar.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat di mana kita akan menampilkan sprite.
Buffer = Alamat memori tempat sprite disimpan.

Procedure : Put1Col (Var PicBuff;X1,Y1:Integer;Col:Byte;Var Buffer);
Fungsi : Menampilkan sprite dengan satu warna.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat tempat sprite akan ditampilkan,
Col = Warna dari sprite.
Buffer = Alamat dari sprite pada memori.

Procedure : Put3D (Var PicBuff;X,Y : Integer;XL,YL:Word; Var Buffer);
Fungsi : Menampilkan sprite yang dapat di-resize.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat di mana sprite akan ditampilkan.

XL,YL = Ukuran baru dari sprite yang diminta.
Buffer = Alamat memori di mana sprite disimpan.

Procedure : T_PutImage (Var PicBuff;X1,Y1:Integer;Var Buffer;
Var PalTable);
Fungsi : Menampilkan sprite secara transparant.
Parameter : PicBuff = Alamt virtual screen.
X1, Y1 = Koordinat di mana sprite akan ditampilkan.
Buffer = Alamat memori di mana sprite disimpan.
PalTable = Alamat memori dari tabel transparant.

Procedure : T_Put1Col (Var PicBuff;X1,Y1:Integer;Col:Byte;Var Buffer;
VarPalTable);
Fungsi : Menampilkan sprite secara transparant dan satu warna.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat di mana sprite akan ditampilkan.
Col = Warna dari sprite.
Buffer = Alamat dari sprite.
PalTable = Alamat dari tabel transparant.

Procedure : T_Put3D (Var PicBuff;X,Y : Integer;XL,YL:Word;
Var Buffer;VarPalTable);
Fungsi : Menampilkan sprite dengan teknik resize dan transparant.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat di mana sprite akan
ditampilkan.
XL, YL = Ukuran baru dari sprite yang diminta.
Buffer = Alamat di mana sprite disimpan.
PalTable = Alamat dari tabel transparant.

Procedure : WordBlockImage (Var PicBuff;X,Y : Integer;Var Buffer);
Fungsi : PutImage dengan sistem WORD dengan tidak mempedulikan
warna 0 dan pembatasan layar.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat dari sprite yang akan ditampilkan,
Buffer = Alamat dari sprite.

Procedure : ByteBlockImage (Var PicBuff;X1,Y1:Integer;Var Buffer);
Fungsi : PutImage dengan sistem BYTE dengan tidak mempedulikan
warna 0.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat dari sprite yang akan ditampilkan,
Buffer = Alamat dari sprite.

Procedure Transparansi

Procedure : Install_Transparent(PalTable:Pointer;Name:String);
Fungsi : Membaca sebuah tabel transparant dari file.
Parameter : PalTable = Alamat dari tabel transparant.
Name = Nama file tabel transparant.

Procedure : Search_Transparent_Palette(Name:String);
Fungsi : Menciptakan sebuah tabel transparant dan menyimpannya ke dalam sebuah file.
Parameter : Name = Nama dari file yang akan digunakan untuk menyimpan tabel transparant.

Procedure Struktur Palette

Procedure : SetVisualPalette (No,R,G,B:Byte);
Fungsi : Mengubah susunan RGB dari sebuah palette yang aktif.
Parameter : No = Nomor Palette yang akan dirubah.
R,G,B = Intensitas Red, Green dan Blue yang baru.

Procedure : GetVisualPalette (No:Byte; Var R,G,B:Byte);
Fungsi : Membaca susunan RGB dari sebuah palette yang aktif.
Parameter : No = Nomor Palette yang akan diambil.
R,G,B = Output dari Red, Green dan Blue yang terbaca.

Procedure : SetPalette (Var Arr_Pal:Palette_Addr; No,Rr,Gg,Bb:Byte);
Fungsi : Mengubah susunan RGB dari sebuah palette pada virtual palette.
Parameter : Arr_Pal = Alamat dari virtual palette.
No = Nomor palette yang akan diubah.
Rr,Gg,Bb = Intensitas Red, Green dan Blue yang baru.

Procedure : GetPalette (Var Arr_Pal:Palette_Addr; No:Byte;
Var Rr,Gg,Bb:Byte);
Fungsi : Membaca susunan RGB dari sebuah palette pada virtual palette.
Parameter : Arr_Pal = Alamat dari virtual palette.
No = Nomor palette yang akan dibaca.
Rr,Gg,Bb = Output dari Red, Green, Blue yang terbaca.

Procedure : Load_Palette (Pal_Buff:Pointer; File_Name:String);
Fungsi : Membaca susunan palette dari file.

Parameter : Pal_Buff = Alamat dari virtual palette yang akan digunakan untuk menyimpan susunan palette.
File_Name = Nama file tempat susunan palette disimpan.

Procedure : Save_Palette (Pal_Buff:Pointer; File_Name:String);
Fungsi : Menyimpan virtual palette ke dalam file.
Parameter : Pal_Buff = Alamat dari virtual palette yang akan disimpan.
File_Name = Nama file tempat virtual palette akan disimpan.

Procedure : Refresh_Palette;
Fungsi : Mengaktifkan palette pada virtual palette Active_Pal ke layar.

Procedure : Active_Palette (Ac_Pal:Palette_Addr);
Fungsi : Memindahkan isi virtual palette ke dalam Active_Pal.
Parameter : Ac_Pal = Alamat virtual palette yang akan dipindahkan ke Active_Pal.

Procedure : Restore_Palette;
Fungsi : Memindahkan isi Default_Pal ke dalam Active_Pal dan sekaligus mengaktifkan Active_Pal ke layar.

Procedure : Install_Palette (Name:String);
Fungsi : Membaca virtual palette dari file dan memasukannya ke dalam Default_Pal.
Parameter : Name = Nama dari file yang akan dibaca.

Procedure Manipulasi Palette

Procedure : Set_All_Palette (Pal_Buff:Palette_Addr; R,G,B : Byte);
Fungsi : Mengisi semua palette pada virtual palette dengan RGB yang diberikan.
Parameter : Pal_Buff = Alamat dari virtual palette.
R, G, B = Intensitas yang akan diisikan.

Procedure : Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Fungsi : Mengubah intensitas Active_Pal secara perlahan ke Pal_Dest dengan delay DeL.
Parameter : Pal_Dest = Alamat dari virtual palette tujuan.
Del = Delay dari loop.

Procedure : Metamorph_Palette(Pal_Buff, Pal_Dest : Palette_Addr;
Rate:Byte) : Boolean;
Fungsi : Mengubah intensitas Pal_Buff ke Pal_Dest sebanyak Rate.

Parameter : Pal_Buff = Virtual palette yang akan diubah.
Pal_Dest = Virtual palette tujuan.
Rate = Jumlah perubahan yang dilakukan.
Return : TRUE jika seluruh intensitas Pal_Buff telah sama dengan Pal_Dest.

Procedure Scroll Layar

Procedure : Load_Vert_Scroll (Name:String);
Fungsi : Membaca background page yang akan dipergunakan untuk vertikal scrolling.
Parameter : Name = Nama dari file background page.

Procedure : Free_Vert_Scroll;
Fungsi : Melenyapkan background page.

Procedure : Put_Vert_Scroll (Var PicBuff; Y:Integer);
Fungsi : Meletakkan verticall scroll pada virtual screen.
Parameter : PicBuff = Alamat virtual screen.
Y = Koordinat Y sprite dari background page sebagai start awal dari penggambaran vertikal scroll.

Procedure : Load_Horiz_Scroll (Name:String);
Fungsi : Membaca background page yang akan dipergunakan untuk horizontal scrolling.
Parameter : Name = Nama dari file background page.

Procedure : Free_Horiz_Scroll;
Fungsi : Melenyapkan background page.

Procedure : Put_Horiz_Scroll (Var PicBuff;Y:Integer);
Fungsi : Meletakkan horizontal scroll pada virtual screen.
Parameter : PicBuff = Alamat dari virtual screen.
Y = Koordinat X sprite dari background page sebagai start awal dari penggambaran horizontal scroll.

Procedure : Load_Parallax_Scroll (Name:String);
Fungsi : Membaca background page yang akan dipergunakan untuk parallax scrolling.
Parameter : Name = Nama dari file background page.

Procedure : Put_Parallax_Scroll(X,Y:Integer;Var PicBuff;Var Temp);
Fungsi : Meletakkan parallax scroll pada virtual screen.
Parameter : X,Y = Koordinat X dan Y sprite dari background page sebagai start awal dari penggambaran parallax scroll.
PicBuff = Alamat dari virtual screen.
Temp = Alamat memori tempat dilakukannya proses unkompres oleh proses parallaxing.

Procedure Kompresi Sprite

Procedure : Color_Compress (Var Source;Var Destination;Var Size:Word);
Fungsi : Mengkompres data dengan metode Color Compress.
Parameter : Source = Alamat data yang akan dikompres.
Destination = Alamat tujuan.
Size = Output besar data setelah dikompres.

Procedure : Color_UnCompress (Var Source;Var Destination);
Fungsi : Meng-unkompres data dengan metode Color Compress.
Parameter : Source = Alamat data yang akan di-unkompres.
Destination = Alamat tujuan.

Procedure : Zero_Compress (Var Source;Var Destination;Var Size:Word);
Fungsi : Mengkompres data dengan metode Zero Compress.
Parameter : Source = Alamat data yang akan dikompres.
Destination = Alamat tujuan.
Size = Output besar data setelah dikompres.

Procedure : Zero_UnCompress (Var Source;Var Destination);
Fungsi : Meng-unkompres data dengan metode Zero Compress.
Parameter : Source = Alamat data yang akan di-unkompres.
Destination = Alamat tujuan.

Procedure : Rll_Compress (Var Source;Var Destination;Var Size:Word);
Fungsi : Mengkompres data dengan metode RLL Compress.
Parameter : Source = Alamat data yang akan dikompres.
Destination = Alamat tujuan.
Size = Output besar data setelah dikompres.

Procedure : RLL_UnCompress (Var Source,Destination);
Fungsi : Meng-unkompres data dengan metode RLL Compress.
Parameter : Source = Alamat data yang akan di-unkompres.
Destination = Alamat tujuan.

Procedure Pembantu

Procedure : GetAddr(Var PicBuff;Var Seg,Offs:Word);
Fungsi : Mengetahui alamat offset dan segment dari sebuah alamat pointer.
Parameter : PicBuff = Alamat pointer yang akan diperiksa.
Seg = Output Segment dari alamat pointer tersebut.
Offs = Output Offset dari alamat pointer tersebut.

polygon bank, 106
polygon shading, 95
port, 27
pre-order multiplication, 239
program
proyeksi titik dengan parallel projection, 4
proyeksi balok dengan parallel projection, 7
proyeksi titik dengan perspective projection, 13
proyeksi balok dengan perspective projection, 15
merancang obyek dengan teknik vektor, 36
perhitungan vektor dan matrix, 58
fungsi transformasi pada MATH.TPU, 75
transformasi objek, 87
proyeksi wireframe menggunakan polygon shading, 100
transformasi objek dengan polygon shading, 118
transformasi objek dengan filter BackFace, 143
transformasi objek dengan painter sort, 164
transformasi objek dengan filter view volume, 182
transformasi objek dengan flat shadded ambient light sourcing, 197
proyeksi menggunakan sudut pandang camera, 213
flight simulator, 241, 290
proyeksi tiga dimensi ke dua dimensi, 3

R
rotasi, 23

S
simetris, 29
sinus, 64, 280
skalar, 45
sprite, 30
sumbu koordinat, 19

T
tangent, 64
teknik vector, 33
tiga dimensi, 2
titik origin, 18
tombol fungsi, 79
transformasi, 46
transformasi
dasar-dasar, 19

U
unit vektor, 55

V
vektor, 45, 54
vertex, 97
viewport, 225
view volume, 19, 182
volume filter, 178
VGACAD.PAS, 317

W
wireframe projection, 34

KUNJUNGI ELEX MEDIA KOMPUTINDO online DI CYBERSPACE
http://www.elexmedia.com

INFORMASI GRATIS DARI PENERBIT
BUKU KOMPUTER BERMUTU

Anda akan mendapatkan sesuatu yang bermanfaat setelah mengunjungi kami

- **katalog online** — informasi buku yang tepat untuk Anda
- **daftar buku terbaru** — informasi buku terbaru
- **rencana STO** — informasi produk baru PT Elex Media Komputindo bagi penjual
- **artikel menarik** — tip dan trik menggunakan komputer dan penanganan masalah komputer
- **Mikrodata Online** — media hobiis dan programmer komputer
- **Audiophile Online** — media hobiis audio (diupdate setiap Kamis)
- **Internet Online** — media penjelajah internet (diupdate setiap Sabtu)

Anda dapat bertanya, mengkritik, memuji, memberi pesan dan kesan terhadap produk PT Elex Media Komputindo melalui e-mail kami: **elex@elexmedia.com**

Anda tertarik dengan Internet?
Baca buku-buku Internet terbitan kami

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional 3

Dunia pemrograman game makin berkembang pesat dengan didukung oleh processor yang semakin cepat dan disertai dengan munculnya hardware-hardware baru yang mendukung graphic tiga dimensi. Sehingga banyak bermunculan game-game, seperti DUKE NUKEM 3D, QUAKE dan lain-lain.

Pada kesempatan ini akan dibahas mengenai 3D PROGRAMMING. Di sini akan dijelaskan bagaimana merancang sebuah FLIGHT SIMULATOR yang didukung dengan teknik pencahayaan (Light Sourcing), teknik-teknik tranformasi dan filter-filter tiga dimensi. Pada bagian terakhir akan dijelaskan bagaimana meng-optimize (mempercepat) sebuah game tiga dimensi sehingga memiliki kecepatan yang memadai untuk dimainkan.

Compiller yang digunakan masih tetap Turbo Pascal 7.0 yang dibantu dengan routine Assembler.

Untuk lebih jelasnya, silakan membaca buku ini.

Penerbit PT Elex Media Komputindo
Jl. Palmerah Selatan 22
Jakarta 10270
Web Page: http//www.elexmedia.com

Kelompok
Pemrograman
Ketrampilan
☑ Tingkat Pemula
☑ Tingkat Menengah
☐ Tingkat Mahir
Jenis Buku
☐ Referensi
☑ Tutorial
☐ Latihan

ISBN 979-637-340-8
12195321